applied sciences

IMPACT FACTOR 2.5

CITESCORE 5.5

Article

A Benchmarking Framework for Hybrid Quantum–Classical Edge-Cloud Computing Systems

Guoxing Yao and Lav Gupta

MDPI

https://doi.org/10.3390/app151810245

Article

A Benchmarking Framework for Hybrid Quantum–Classical Edge-Cloud Computing Systems

Guoxing Yao and Lav Gupta *

Department of Computer Science, University of Missouri-St. Louis, St. Louis, MO 63121, USA; gym8g@umsystem.edu
* Correspondence: lavgupta@umsl.edu

Abstract

Quantum computers are emerging as a major tool in the computation field, leveraging the principles of quantum mechanics to solve specific problems currently beyond the capability of classical computers. This technology holds significant promise in edge-main cloud deployments, where it can enable low-latency data processing and secure communication. This paper aims to establish a research foundation by integrating quantum computing with classical edge-cloud environments to promote performance across a range of applications that scientists are actively investigating. However, the successful deployment of hybrid quantum–classical edge-clouds requires a comprehensive evaluation framework to ensure their alignment with the performance requirements. This study first proposes a novel quantum benchmarking framework, including two distinct methods to evaluate latency scores based on the quantum transpilation levels across different quantum-edge-cloud platforms. The framework is then validated for the edge-cloud environment by benchmarking several well-known and useful quantum algorithms potentially useful in this domain, including Shor's, Grover's, and the Quantum Walks algorithm. An optimal transpilation level is eventually suggested to achieve maximum performance in quantum-edge-cloud environments. In summary, this research paper provides critical insights into the current and prospective capabilities of QPU integration, offering a novel benchmarking framework and providing a comprehensive assessment of their potential to enhance edge-cloud performance under varying parameters, including fidelity and transpilation levels.

Keywords: benchmarking; quantum computing; quantum-edge-cloud; hybrid quantum edge

Academic Editor: Nikos Konofaos

Received: 28 July 2025
Revised: 8 September 2025
Accepted: 16 September 2025
Published: 20 September 2025

Citation: Yao, G.; Gupta, L. A Benchmarking Framework for Hybrid Quantum–Classical Edge-Cloud Computing Systems. *Appl. Sci.* **2025**, *15*, 10245. https://doi.org/10.3390/app151810245

1. Introduction

Although still in its early stages, quantum computing is widely regarded as a transformative development in computational technology, offering capabilities far beyond those of classical systems. Unlike classical computers, which exist in mutually exclusive states (0 or 1), quantum computers use quantum bits, or qubits, which can exist in a superposition of states, enabling them to represent and process multiple possibilities simultaneously. In contrast, quantum computers utilize quantum bits (qubits), which can exist in superposition, representing multiple states simultaneously [1]. This fundamental difference enables quantum systems to process complex problems more efficiently, particularly those involving high-dimensional search spaces and probabilistic reasoning.

Superposition is a defining feature of quantum systems that allows each qubit to simultaneously occupy multiple potential states, each associated with a particular probability of being observed. However, while qubits can exist in multiple superposition states, they collapse to one of two possible physical states represented as ∣0> or ∣1> when measured or even because of accidental interaction with the environment. This probabilistic nature, in contrast to the deterministic behavior of classical bits, enables quantum computers to process vast amounts of information in parallel [2]. Superposition is also essential for quantum entanglement, another core principle in which two or more qubits become correlated such that the state of one directly determines the state of the other, regardless of physical separation [3]. Entangled states are special connections between two quantum particles, irrespective of their separation. For example, in the case of the Bell state $|\Phi+\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$, if one particle is in state "0", the other will also be in state "0" and if the first one is in state "1", the other will also be in "1". This powerful feature underpins several applications like quantum teleportation, secure communication, and distributed quantum computing.

Together, superposition and entanglement enable quantum systems to outperform classical architectures in promising breakthroughs in areas such as cryptography, optimization, machine learning, and quantum networking [4,5]. However, current quantum hardware—often referred to as Noisy Intermediate-Scale Quantum (NISQ) devices—is limited by challenges such as short coherence times, gate errors, and sensitivity to environmental noise. These limitations become even more pronounced when integrating quantum processing units (QPUs) into edge-cloud environments, where factors like network latency, synchronization, and bandwidth must also be considered.

Despite these challenges, research on hybrid quantum–classical edge-cloud systems is gaining momentum. This progress is supported by the increasing availability of both simulated and real quantum platforms such as IBM Quantum Lab and Amazon Braket. To enable the effective integration of QPUs in such new environments, a robust benchmarking framework is essential [6,7]. Such a framework provides a consistent way to measure, compare, and evaluate the performance of different platforms, helping researchers choose the most suitable option for their applications [8,9]. Recent advances in quantum platforms have accelerated the development of benchmarking methodologies that assess and compare quantum hardware based on consistent performance indicators, including qubit count, gate fidelity, coherence time, and circuit depth derived from executing representative algorithms. In edge-cloud contexts, additional metrics such as transpilation latency, execution time, and communication overhead are equally critical [10]. While some of these are hardware and network dependent, they can still be characterized systematically by defining a standardized benchmarking framework. This work attempts to achieve this by selecting a common set of algorithms, controlled conditions, and statistical characterization.

To support this emerging ecosystem, this paper proposes a comprehensive novel benchmarking framework tailored to assess the performance of hybrid quantum–classical edge-cloud environments. The framework evaluates computational performance using a suite of canonical quantum algorithms—Shor's [11], Grover's [12], and Quantum Walks [13]—all known for their high computational intensity and classical intractability. By testing these algorithms on both simulated and real quantum hardware across multiple platforms, including IBM Quantum Lab [14] and Amazon Braket [15,16], the framework enables a thorough comparison based on practical execution factors like gate fidelity, transpilation overhead, and latency.

This research makes the following key contributions:

(1) It proposes a novel benchmarking framework for evaluating quantum systems within edge-cloud networks using real-world performance metrics.

(2) It leverages established quantum algorithms to assess the performance of quantum platforms under varied conditions and computational loads.

(3) It implements an auxiliary communication model within a virtualized environment to emulate realistic network latencies, enabling comprehensive evaluation of communication performance.
(4) It presents insights from longitudinal experiments, revealing platform-specific trends and integration challenges, particularly in terms of optimizing the integration of quantum technology into practical, real-world tasks.
(5) It offers performance optimization strategies derived from value analysis, facilitating informed tuning of system parameters to enhance efficiency.

By establishing a standardized and reproducible benchmarking approach, this research provides a foundational tool for evaluating quantum systems in distributed environments and supports progress toward achieving practical quantum advantage in real-world cloud scenarios.

The remainder of the paper is organized as follows: Section 2 reviews the conceptual background underlying the research. Section 3 discusses related work and identifies research gaps. Section 4 details the materials and methods used. Section 5 describes the experimental setup and outcomes, while Section 6 provides a comprehensive analysis of results and conclusions. Finally, Section 7 outlines directions for future research.

2. Background and Conceptual Framework

This section provides a quick review of the concepts necessary to appreciate the discussion on the proposed benchmarking framework and its experimental evaluation.

2.1. Fundamental Mechanisms in Quantum Computing

Quantum mechanical phenomena are inherently challenging to comprehend due to their departure from classical intuition and the inapplicability of everyday macroscopic experiences. Quantum mechanics constitutes a mathematical framework characterized by a set of foundational axioms, from which the behavior of quantum systems is logically derived [17].

2.1.1. Dirac Notation, Superposition, Parallelism, and Entanglement

Quantum state spaces, along with the transformations that act on them, are typically formulated using linear algebraic structures such as vectors, matrices, and Dirac's bra-ket notation. These state vectors represent particles such as electrons or polarized photons [18] and reside in an n-dimensional complex Hilbert space [18]. Today, most physicists use Dirac notation for its conciseness and clarity [19]. For example, a quantum state vector is denoted as a ket, $|q\rangle$, and its conjugate transpose (the dual vector) is written as a bra, $\langle q|$. A typical ket $|q\rangle$ is often a superposition of two basis vectors, $|0\rangle$ and $|1\rangle$, with complex probability amplitudes α and β. Mathematically, it is expressed as

$$|q\rangle = \alpha\ |0\rangle + \beta\ |1\rangle \tag{1}$$

with the normalization condition

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2}$$

In quantum computing, the ket $|q\rangle$ represents a qubit. These quantum states scale exponentially following a growth factor of 2^n, where n is the number of qubits. This scaling is typically represented using the tensor product, denoted by the symbol $\otimes$ [20]. As a result, quantum computers can represent and process information in ways that far exceed the capabilities of classical computers.

The second foundation mechanism is named quantum parallelism [21]. This refers to the ability of a quantum computer to use superposition to process all possible input states simultaneously [17]. Superposition allows qubits to exist in a combination of states, allowing quantum computers to evaluate multiple computational paths at once, thereby accelerating certain computations. Shor's algorithm is a prominent example that elegantly exploits quantum parallelism using quantum Fourier transform [17]. This differs fundamentally from classical parallelisms, in which operations are performed on mutually exclusive states [21].

The third important aspect of quantum computing is entanglement, or entangled quantum states. A quantum state $|q\rangle$ is called entangled if it cannot be expressed as a simple tensor product of two individual qubit states. In the context of measurement, two particles are considered unentangled if the measurement outcome of one does not influence the outcome of the other [16]. In contrast, the entangled qubits are correlated in such a way that the state of one qubit directly influences the state of the other, regardless of the distance between them.

Through these three foundational mechanisms, superposition, quantum parallelism, and entanglement—quantum computing can perform computations that are infeasible for classical computers and handle scaled computing tasks beyond the theoretical capacity of classical computers, offering the possibility of solving problems previously considered intractable.

2.1.2. Gate-Based Quantum Computing

Building on the principles discussed in Section 2.1, quantum gates act as operators on kets $|q\rangle$, which serve as operands in quantum computations. These gates perform linear transformations on quantum states, and to preserve both orthogonality and the norm state of vectors, these transformations must be unitary. Mathematically, quantum computing involves applying a series of unitary transformations to an initial quantum state, typically represented as $|q_0q_1...q_n\rangle$. Unlike classical logical gates, most quantum gates, such as Hadamard, Pauli, and CNOT gates, are inherently unitary gates since they meet the requirement of preservation of orthogonality and reversibility [17]. A gate is unitary if it can be represented by a unitary matrix; its inverse is equal to its conjugate transpose. Eventually, the quantum state is measured in quantum computing, collapsing the superposition into a classical outcome.

2.1.3. Selected Quantum Devices in Hybrid Edge Architectures

In a hybrid quantum–classical edge-cloud system, quantum devices are distributed across the network according to their capabilities. While placement is flexible, devices in higher layers (e.g., the main cloud) generally provide greater computational or communication capabilities than those at the edge or endpoints [22]. Below is an overview of the key quantum devices across these layers:

1. Quantum processing unit (QPU): QPU is the core computational component of a quantum computer, executing quantum circuits on qubits to solve complex problems that are intractable for classical computers. The number of physical qubits in current gate-based quantum computers, particularly superconducting qubits (used by IBM and Google) and trapped-ion qubits (used by Quantinuum), is currently less than 1500. In such a scenario, algorithms that distribute workload over classical and quantum resources are essential to enhance reliability and performance.
2. Quantum Switches (QS): quantum switches enable connection of channels between devices in a quantum network for routing quantum information or distributing

entanglements. A quantum switch can handle several quantum channels at the same time [23].

3. Quantum Repeaters (QR): these devices extend communication distance by overcoming signal loss without violating the no-cloning principle, using techniques such as teleportation, entanglement swapping, and error correction [24]. Quantum relays function similarly but lack entanglement purification and quantum memory, instead improving transmission range by boosting the signal-to-noise ratio at detectors [25].
4. Quantum memory: quantum memory stores states of single photons or entangled states for short periods, without destroying their quantum properties. They assist hybrid edge computing by enabling synchronization of operations and storing entangled states until nodes are aligned.
5. Quantum sensors: utilize properties like superposition and entanglement to achieve measurement capabilities exceeding those of classical sensors [26]. At the edge, quantum sensors provide ultra-precise measurements and improve the quality of input data. These sensors can operate reliably in constrained environments like battlefields and remote areas.

2.1.4. Architectural Aspects of the Hybrid Edge-Cloud Systems

Quantum–classical hybrid systems combine the computational power of quantum processors with the reliability, scalability, and flexibility of classical systems. Given that the current quantum devices belong to the NISQ era, characterized by a limited number of qubits, short coherence times, and significant noise, classical systems play a vital role in complementing their limitations. They support essential tasks such as data preprocessing, iterative feedback, and error mitigation.

In these hybrid environments, quantum processors are typically tasked with solving computationally intensive problems, such as optimization, simulation, and cryptography, while classical systems handle peripheral operations. The synergy between the two paradigms enables problem-solving capabilities that neither system could achieve alone. This collaboration is especially relevant for complex, large-scale problems like supply chain logistics or quantum chemical simulations.

Modern applications are inherently hybrid, requiring both quantum and classical resources to function effectively. Classical components handle preprocessing to generate circuits for state preparation and attach these to quantum algorithms. Classical components also handle postprocessing, such as the correction of readout errors by applying an unfolding method to compute the undisturbed result distribution from the disturbed measured distribution. For instance, even quantum algorithms like Shor’s require classical postprocessing to complete factorization [27]. To support this level of integration, hybrid edge-cloud architecture distributes diverse computational resources, ranging from IoT devices to cloud-accessible quantum hardware, across edge and cloud environments. A dedicated orchestration layer coordinates the system, dynamically assigning tasks to classical or quantum components based on predefined policies or intelligent optimization mechanisms.

Figure 1 shows the layered hierarchical architecture of hybrid systems organized into five distinct layers:

1. End-point Layer: this layer comprises sensing, actuating, and measuring devices located near the physical environment. These devices collect high volumes of raw data, which must be processed or forwarded for analysis.
2. Edge Layer: the edge layer includes local classical compute nodes with moderate processing and storage capacity. Quantum devices may also be integrated at the edge to support latency-sensitive hybrid computations closer to the data source.

3. Main Cloud Layer: this layer provides access to high-capacity classical and quantum computing resources for computationally intensive tasks, such as training large-scale AI models or executing complex quantum circuits. Cloud-based quantum platforms such as Amazon Braket, IBM Quantum, and Quantinuum Nexus allow users to compile, transpile, and optimize quantum programs through cloud APIs.
4. Communication layer: this layer provides reliable and secure transmission channels created through classical and quantum communication processors, routers, switches, relays, and repeaters. This layer aims to maintain low latency and high throughput.
5. Orchestration layer: this layer is responsible for managing and coordinating cross-layer provisioning of resources from a pool of interacting classical and quantum components. It carries out key functions such as task partitioning, scheduling, and workflow management across the hybrid environment. It helps adapt to system challenges by handling noise and errors, device heterogeneity, and communication across components.

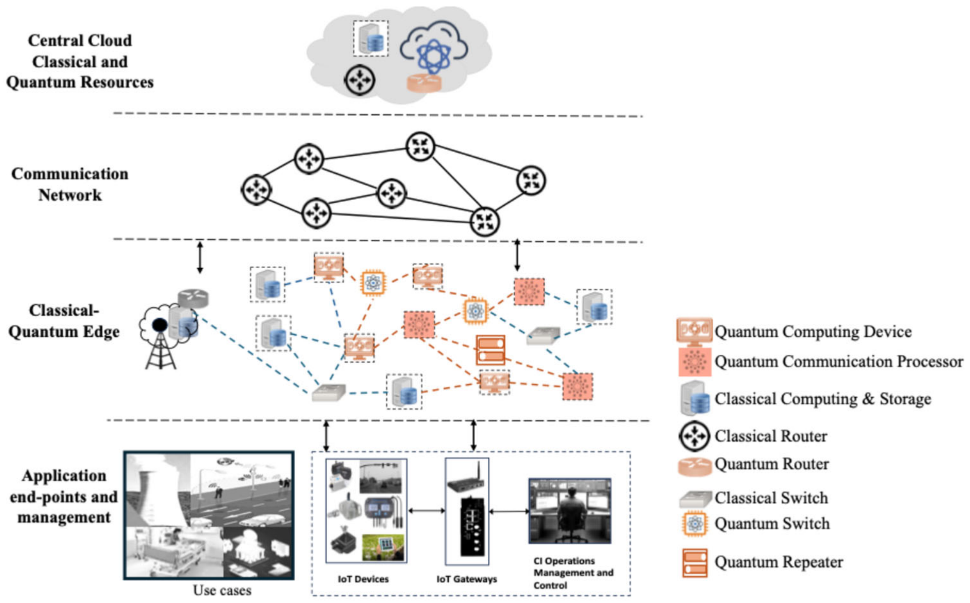
Central Cloud Classical and Quantum Resources
Communication Network
Classical-Quantum Edge
Application end-points and management
Use cases
IoT Devices
IoT Gateways
CI Operations Management and Control
Quantum Computing Device
Quantum Communication Processor
Classical Computing & Storage
Classical Router
Quantum Router
Classical Switch
Quantum Switch
Quantum Repeater

Figure 1. Layered and hierarchical architecture of hybrid quantum–classical edge-cloud systems (orchestration layer not shown).

2.2. Background of Quantum Benchmarking

The current state of quantum computing is said to be Noisy Intermediate-Scale Quantum (NISQ), characterized by quantum processors that can execute computations intractable for classical computers but are limited by noise, finite coherence time, and a restricted number of qubits. In this context, quantum benchmarking assumes importance as researchers attempt to evaluate and compare quantum devices to inform future developments [28]. The availability of small gate-based quantum computers, such as those provided by Amazon Bracket and IBM Quantum platforms, has accelerated the development of quantum benchmarking practices. These platforms allow researchers to test and compare the performance of different quantum computers by running various quantum algorithms, helping them see how well a device performs against other devices or against expected results from theory and classical simulations [10].

A gate-based quantum device processes input qubits by performing unitary transformations through a structured sequence of gate operations and measurements, represented by a quantum circuit. Simple gate sequences, such as repeated application of CNOT gates, are often used to evaluate the reliability and performance of these circuits. Unlike traditional low-level hardware metrics, algorithmic benchmarking has emerged as a prominent methodology for evaluating quantum computational performance. This approach involves running full algorithms on quantum devices and assessing their correctness, execution time, and robustness. It provides a more application-relevant assessment of system performance than assessment of individual gate fidelities.

2.2.1. Formal Explanation of Quantum Benchmarking

In classical computer science, benchmarking has traditionally relied on a broad consensus rather than a universally agreed definition. It is broadly understood as the process of running standardized tests to evaluate, using relevant metrics, how well a system performs, usually by comparing it against other systems or a reference standard. This conceptual framework has carried over into quantum computing, but quantum benchmarking would bring its own set of challenges and considerations. It needs to consider unique phenomena like entanglement, decoherence, and probabilistic measurement while measuring performance.

While the terms quantum benchmarks and quantum metrics are sometimes used interchangeably, they are distinct. A quantum metric is a quantitative measure of performance designed to assess specific aspects of the performance of a quantum device or a system, such as fidelity or error rate. A benchmark, on the other hand, is a broader test that may involve one or more such metrics.

In general, quantum benchmarking involves testing how well a quantum computing system or its components perform using known algorithms and predefined metrics. It helps researchers and developers to assess their accuracy, reliability, and computational capabilities under real or simulated conditions. Benchmarking also makes it easier to compare different quantum platforms or track performance improvements over time, which is essential as the field continues to evolve [29].

Unlike classical computing benchmarking, which often uses general-purpose suites like LINPACK and SPEC (Standard Performance Evaluation Corporation, Gainesville, VA, USA), quantum benchmarking is inherently purpose-driven. For instance, it may be used to assess the performance of quantum processors within hybrid edge-cloud environments or to evaluate specific hardware components such as qubits, quantum gates, or circuits and devices. The usefulness of the results depends heavily on how well they align with the original intent of the test [30].

At this stage in quantum computing, where systems are still error-prone and limited in scale, benchmarks mostly serve to measure progress and identify bottlenecks. Rather than evaluating full-scale computational utility, most current benchmarks focus on the impact of noise, error rates, and hardware limitations [31].

This paper focuses specifically on benchmarking of gate-based quantum computers, where systems are built using well-understood components like qubits, gates, and circuits. This allows us to create benchmarks that are interpretable and meaningful. The next subsection reviews common benchmarking techniques to identify those relevant for quantum computing in edge-cloud environments.

2.2.2. Types of Quantum Benchmarking

Benchmarking in quantum computing, much like in classical computing, relies on measurable performance metrics to compare systems over time, across technologies, and under different conditions. But benchmarking quantum systems is much more complex due to their inherent noise, probabilistic behavior, and hardware diversity. No single benchmark can universally evaluate quantum hardware, and multiple metrics are often used. Similarly, in the evaluation of hybrid quantum edge-cloud computing, effective benchmarking needs to be multifaceted, addressing many layers of the computing stack with a suite of protocols that target specific performance aspects relevant to the intended application. To better understand where our benchmarking method fits, it helps to investigate the key layers in the quantum benchmarking stack:

1. Qubit-Level Benchmarking focuses on the basic building blocks of quantum gates, i.e., qubits. The number of qubits directly relates to the computational capacity of a quantum processor [32]. The key metrics are coherence times, which quantify the duration over which the qubits maintain their quantum states, and frequency response, which characterizes the qubits' spectrum behavior and interaction with control signals.
2. Gate-Level Benchmarking looks at how well individual quantum gates perform. It is widely recognized as a critical component of any comprehensive quantum benchmarking framework. Common techniques at this layer include
    - Randomized benchmarking (RB): provides an estimate of average gate fidelity while mitigating the influence of SPAM (state preparation and measurement) errors. RB is scalable and is less affected by noise.
    - Direct Fidelity Estimation (DFE): estimates gate fidelity under the assumption of negligible SPAM errors. DFE provides a single value between 0 and 1 as a measure of gate implementation, where 1 represents the ideal gate implementation.
    - Quantum Set Tomography (QPT): offers complete gate characterization but is resource intensive and does not scale well.

Since quantum platforms rely on both high-level gates (used in algorithm design) and native gates (specific to hardware), measuring gate fidelity, circuit depth, and gate density (how efficiently gates are packed into a circuit) is essential to understanding performance.

3. Circuit-Level Benchmarking tests full quantum circuits to see how well they perform under real-world conditions. Benchmarks in this category use some of the following metrics to assess how circuits behave as they scale in depth and complexity:
    - Quantum Volume: reflects a system's ability to reliably execute increasingly complex circuits reliably. It captures a combination of qubit count, gate fidelity, connectivity, and compiler performance.
    - Circuit Layer Operations Per Second (CLOPS): measures the execution speed of a quantum processor in executing layers of QV circuits. It is a throughput-oriented performance metric.
    - Cross-entropy benchmarking (XEB): a scalable noise characterization and benchmarking scheme that estimates the fidelity of an n-qubit circuit composed of single- and two-qubit gates.
4. Processor-Level Benchmarking: This layer evaluates the overall performance of quantum processors. It includes system-wide metrics like quantum volume, holistic workload testing, and time-to-solution, which measures how quickly and precisely a device can complete a task.
5. Platform Level Benchmarking: focuses on the broader quantum ecosystem—cloud access, compilers, SDKs, and APIs. Different providers offer unique environments with their own software stacks, like IBM's Qiskit, Google's Cirq, Quantinuum's TKET, the Berkeley Quantum Synthesis Toolkit (BQSKit), Quantum Tool Suite (QTS), and Duke University's Staq—all offer varying degrees of abstraction, performance optimization, and hardware integration [33]. Benchmarking at this level helps compare platforms using tools like Open Quantum Assembly Language (QASMBench) or MQT Bench, which provide consistent tests across different hardware technologies (e.g., superconducting, trapped ion, or photonic qubits).
6. Application-Level Benchmarking is about testing how well the quantum systems handle real-world problems. Frameworks like Quantum computing Application

benchmark (QUARK), Munich Quantum Toolkit (MQT), and workload-specific tests examine performance from an application perspective, looking at how algorithms like Variational Quantum Factoring (VQF) behave on different systems.

7. Full Stack Benchmarking ties everything together, evaluating the entire system—from hardware and circuits up through orchestration, resource management, and latency. This is especially useful in NISQ-era applications involving hybrid quantum–classical systems, where performance at every layer impacts the outcome.

2.2.3. Quantum Benchmarking Metrics

1. Benchmarking quantum systems requires a range of performance metrics to assess their reliability, speed, and computational accuracy. Some of the most used metrics across quantum benchmarking frameworks include:
2. Qubit coherence: Qubit coherence refers to the duration of time for which a qubit can maintain its quantum state before decoherence occurs. Coherence is essential for preserving superposition and entanglement. Coherence time is a critical performance indicator, as environmental disturbances can cause decoherence, leading to loss of quantum information and rendering computations unreliable [34].
3. Average gate fidelity: This metric quantifies the difference between an ideal unitary gate operation and its real-world implementation. High average gate fidelity indicates that a quantum gate behaves closely to its theoretical counterpart, minimizing operational errors [35].
4. Circuit fidelity: Circuit fidelity measures how accurately an entire quantum circuit executes its intended operation. It evaluates the similarity between the final state of the physical system and the ideal (noiseless) state, reflecting the circuit's overall correctness and resilience to noise [36].
5. Error rate: quantum computing error rate refers to the probability that a qubit will change state during computation. Current quantum computers typically have error rates in the range of 0.1% to 1%, which is significantly higher than the error rates in classical computer systems [37].
6. Quantum Volume: is a metric representing the largest square-shaped quantum circuit (where width, or the number of qubits, equals the depth, or the number of gate layers) that a processor can implement successfully. It captures the interplay of qubit count, gate fidelity, connectivity, and compiler efficiency.
7. Qubit readout fidelity: Also known as qubit measurement fidelity, it refers to the error rate associated with qubit readout operations. It reflects the likelihood that the measured output truly represents the actual state of the qubit at the time of measurement.
8. Quantum gate execution time: also referred to as quantum gate speed, it indicates the time needed to perform quantum gate operations. Faster execution times generally enable deeper circuits within limited coherence windows.
9. SPAM fidelity: State Preparation and Measurement Process (SPAM) fidelity refers to how well a quantum computer can initialize qubits into a desired state and accurately read out the qubit's state after computations. High SPAM fidelity is essential for reliable quantum input-output operations [38].

2.3. Benchmarking in Hybrid Quantum–Classical Edge-Cloud Systems

Benchmarking frameworks help assess how effectively tasks are executed across this hybrid landscape, identifying the most efficient configurations and exposing bottlenecks in resource allocation, latency, and orchestration [8,9]. These distributed architectures combine quantum processors (QPUs) and classical compute units (CPUs/GPUs) deployed across edge devices and centralized cloud infrastructure. The goal is to harness quantum computational advantages while maintaining the scalability, control, and responsiveness offered by classical systems. Challenges in such an environment include quantum noise and error correction, latency and bandwidth limits, software stack complexity, security and privacy across distributed systems, and lack of mature frameworks for hybrid orchestration.

In this context, benchmarking evaluates both the computational and communication performance of hybrid systems. On the computational side, key factors include the efficiency of the quantum circuit transpilation, execution fidelity, and the effectiveness of post-processing. On the communication side, latency between the edge and cloud layers is a critical metric, particularly for time-sensitive applications. By integrating quantum capabilities into edge-cloud systems, the goal is to reduce overall latency, improve computational throughput, and enhance scalability, particularly for demanding tasks in areas like optimization, cryptography, and AI.

To measure performance comprehensively, this paper proposes the use of a quantum-edge-cloud latency score—a unified benchmarking metric that reflects the total time taken to complete a computational task initiated by an edge device. When a computation request goes from the IoT domain to the edge-cloud system, the system performs several key operations: first, it transpiles the quantum circuit to match the specific hardware topology; next, it executes the circuit on a quantum processor or a simulator; finally, it returns the result to the IoT gateway [39]. The total time taken by these steps—including request transmission, processing, execution, and response—are incorporated in the latency score. This latency score (in time units) provides an end-to-end assessment of the system's responsiveness. The higher the latency score, the slower and less efficient the benchmarked system, while a lower latency score reflects more efficient processing and/or communication in the edge-cloud continuum. This makes the metric a powerful indicator of the system's ability to handle both computational and communication demands in a dynamic, resource-constrained environment.

Figure 2 illustrates the quantum workflow through the hybrid quantum–classical edge cloud. It is important to include the transpilation step in the benchmarking process to account for practical limitations of real quantum hardware. Steps 1 and 2 involve communication between the device and the edge-cloud system, while Steps 3 and 4 cover circuit processing and execution on the edge-cloud infrastructure. The total time across these four steps defines the quantum-edge-cloud latency score, a key metric for evaluating system responsiveness.

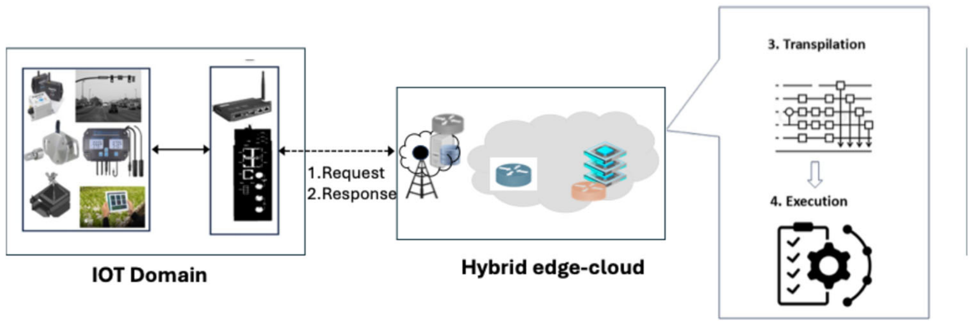

Figure 2. Quantum computation in the hybrid quantum edge-cloud environment.

3. Related Work and Research Gaps

3.1. Related Work

To the best of our knowledge, no existing benchmarking framework addresses the integration of quantum computing within edge-cloud environments. Benchmarking at the system level includes quantum and classical computing and communication components. Most available works focus either on quantum computing or classical edge-cloud networks, without bridging both domains—particularly in terms of end-to-end latency considerations. Within the quantum domain there is application-oriented benchmarking, orchestration-level benchmarking, and architectural aspects of integrated edge-cloud networks. This research reviews some of the relevant research published after 2020 to identify research gaps and reinforce the need for this research.

Research in [40] studies application-oriented benchmark frameworks to assess metrics such as entanglement and path depth. The authors adapt classical benchmarking principles to the quantum domain, offering score calculations and feature vectors to quantify application coverage. Authors in ref. [41] propose an application-oriented benchmark based on quantum that studies the effect of quantum circuit depth limitations on real-world applications. Finzgar et al. propose a framework that focuses specifically on hybrid quantum–classical algorithm evaluation using domain-specific applications like logistics and robotic planning [42]. These benchmarks assume availability of quantum only in the main cloud and omit edge-cloud latency.

The orchestration aspects of hybrid quantum–classical systems have been studied particularly in the context of variational algorithms. The work in [43] demonstrates improvements in execution time and throughput via server-side parameter updates and qubit reset mechanisms. In [44] the research studies the impact of reducing quantum–classical communication overheads to speed up execution of hybrid algorithms [45]. In [46] the authors investigate integrated quantum–classical systems and underscore the need for system-level performance evaluation tools. In [47], the authors introduce a utility-based metric to evaluate the operational benefit of distributing quantum nodes across edge and main clouds, factoring in latency, success rate, and fidelity.

As said above, efforts to benchmark integrated quantum–classical edge-cloud systems are severely limited. For hybrid quantum edge computing networks, the authors in [48] identify latency, edge-cloud synchronization, and device heterogeneity as major challenges. Reference [32] presents MQT Bench, a uniform benchmarking platform supporting multiple abstraction levels with a broad set of benchmarks and cross-level generalizability. Researchers in [41] introduce QASMBench, an end-to-end suite featuring diverse circuit types and evaluation tools for both pre- and post-transpilation analysis, covering small-scale, medium-scale, and large-scale quantum systems. Reference [33] introduces a comprehensive benchmarking suite that evaluates the performance of different mainstream quantum computing software development kits (SDKs). Different types of quantum platforms also work as the category of benchmarks. For instance, references [41,42] focus on platforms like IBM-Q, Rigetti, IonQ, Quantinuum, and AQT@LBN.

3.2. Research Gaps in Quantum Computing, Edge-Cloud Computing, and Quantum Benchmark Framework

The previous section reveals a critical research gap in existing benchmarking approaches, particularly in the context of integrated quantum–classical edge-cloud environments. Some of the issues that have not been adequately focused on are

1. Since the integration of quantum computing and the edge-cloud environment is still in its infancy, the focus of most existing research is on gains on the privacy front rather than performance issues like end-to-end latency involving distributed compute and networking over edge, and cloud domains. The existing benchmarking frameworks like SupermarQ and Quark evaluate centralized hybrid algorithms and do not account for quantum and classical resources distributed at the edge and in the main clouds.
2. Multi-domain architectures are not focused on. Real life environments are envisaged to have classical and quantum devices communicating with each other in the same and across domains.

3. Transpilation and its topology optimization levels are not stressed as important factors. If a quantum computer needs to run high-level circuits, these circuits must be converted (i.e., transpiled) to the corresponding primitive quantum gates of specific QPUs [49]. Unlike traditional compilation, which translates source code into a target language [50], transpilation specifically transforms a quantum circuit to align with the topology and constraints of a given quantum device [50]. For instance, while some compilers provide multiple levels of optimization for classical systems, many quantum platforms also offer varying optimization levels in transpilation. This step is pivotal because the efficient execution of quantum circuits depends on how well the transpilation process adapts the logical qubit connections to the physical architecture of the QPUs.
4. Insufficient evaluation of heterogeneous resource allocation. Benchmarks do not evaluate how well systems handle resource-aware dynamic offloading or adaptive load balancing in hybrid edge-cloud scenarios.

While this paper does not claim to bridge all the gaps indicated above, the authors take care of cross-domain issues, heterogeneity, and transpilation by benchmarking end-to-end latency in the hybrid quantum–classical edge-cloud environment.

4. Materials and Methods

Keeping in view the research gaps discussed in Section 3.2, this research aims to bridge the chasm between quantum computing and edge-cloud networks. For a comprehensive evaluation of hybrid quantum–classical edge-cloud systems, it is important to benchmark real-world quantum parameters like fidelity and optimization levels of transpilation. This research proposes a benchmarking framework that measures and helps control latency related to a multi-domain, distributed, hybrid quantum–classical edge-cloud environment.

4.1. Description of the Benchmarking Framework

This subsection presents the detailed formulation of the proposed benchmarking framework. Building upon the conceptual task flow illustrated in Figure 2 for the IoT–Edge-cloud hybrid classical-quantum environment, the total latency is decomposed into three main components: communication latency, transpilation latency, and execution latency, as depicted in Figure 3.

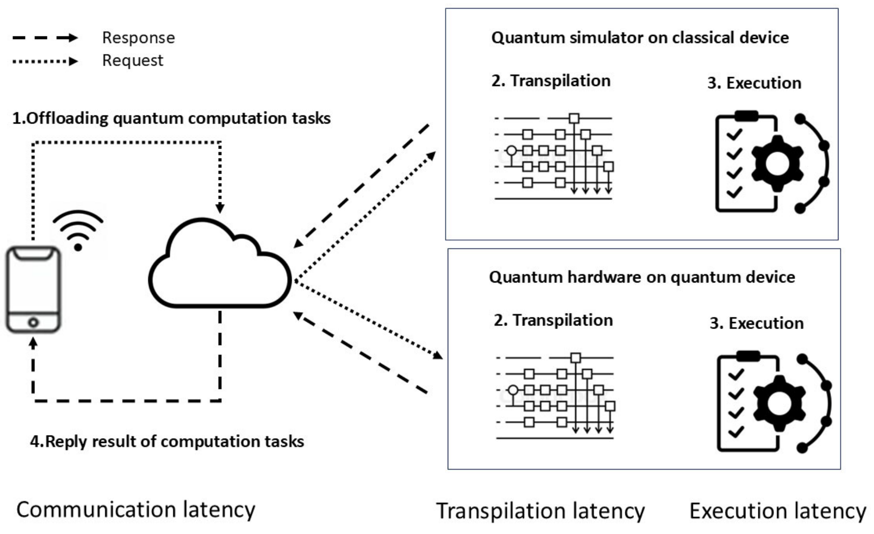
Response
Request
1.Offloading quantum computation tasks
4.Reply result of computation tasks
Quantum simulator on classical device
2. Transpilation
3. Execution
Quantum hardware on quantum device
2. Transpilation
3. Execution
Communication latency
Transpilation latency
Execution latency

Figure 3. Communication, transpilation, and execution latencies in hybrid workflows.

To evaluate the impact of different optimization strategies, two methods are introduced: Method A, which uses predefined ordinal optimization levels (e.g., levels 0–3, as defined by the IBM platform), and Method B, which utilizes continuous decimal values to represent optimization levels more precisely.

This research defines the Quantum–Edge-cloud Latency Score (denoted as L) as a comprehensive metric for evaluating the end-to-end round-trip latency in a hybrid quantum–classical edge-cloud system. This score encompasses all key stages in the data processing pipeline: request transmission from an IoT or edge device, quantum circuit transpilation, quantum circuit execution, and result transmission back to the originating device.

Formally, the total latency, or the L-score is computed as the sum of all latencies from the IoT devices to the main cloud. It can be represented as

$$L = \sum_{i=1}^{n} \tau_i \tag{3}$$

This work is limited to three primary sources of latency that significantly impact the performance of hybrid quantum–classical edge-cloud systems. These components are explained below:

- τ_{cl}: Communication Latency

  This refers to the time required for data to travel between the operational technology (OT) layer, consisting of devices such as sensors and IoT devices, and the edge-cloud servers. It includes a range of delays introduced by network transmission, message routing, protocol handling, network congestion, and hardware-related constraints. These factors collectively influence the responsiveness and throughput of the system.

- τ_{tl}: Transpilation Latency

  This is the time required to convert a high-level quantum circuit into a form compatible with the physical constraints of a specific processor. This process includes the application of predefined transpilation optimization settings. In this step, predefined transpilation optimization parameters are applied. Similarly to classical compilation, higher levels of transpilation optimization result in longer transpilation times.

- τ_{el}: Execution Latency

  This measures the time taken by the edge-cloud server to execute the transpiled quantum circuit on either a QPU or a quantum simulator. As shown as Step 4 in Figure 4, this latency depends on the circuit's complexity, the quality of the transpilation, and the execution performance of the underlying quantum hardware.

In our case, the total latency, or the L-score, can be modeled as the sum of the three components described:

$$L = \tau_{cl} + \tau_{tl} + \tau_{el} \tag{4}$$

The unified L-score provides a composite and robust benchmark for evaluating end-to-end performance of hybrid quantum–classical systems in edge-cloud environments. This metric is particularly useful in applications where low-latency performance is critical, such as real-time analytics, autonomous systems, and edge-based quantum machine learning. A thorough evaluation of the latency score allows researchers and developers to identify performance bottlenecks, optimize system configurations, and enhance the integration of quantum technologies into edge-cloud infrastructures.

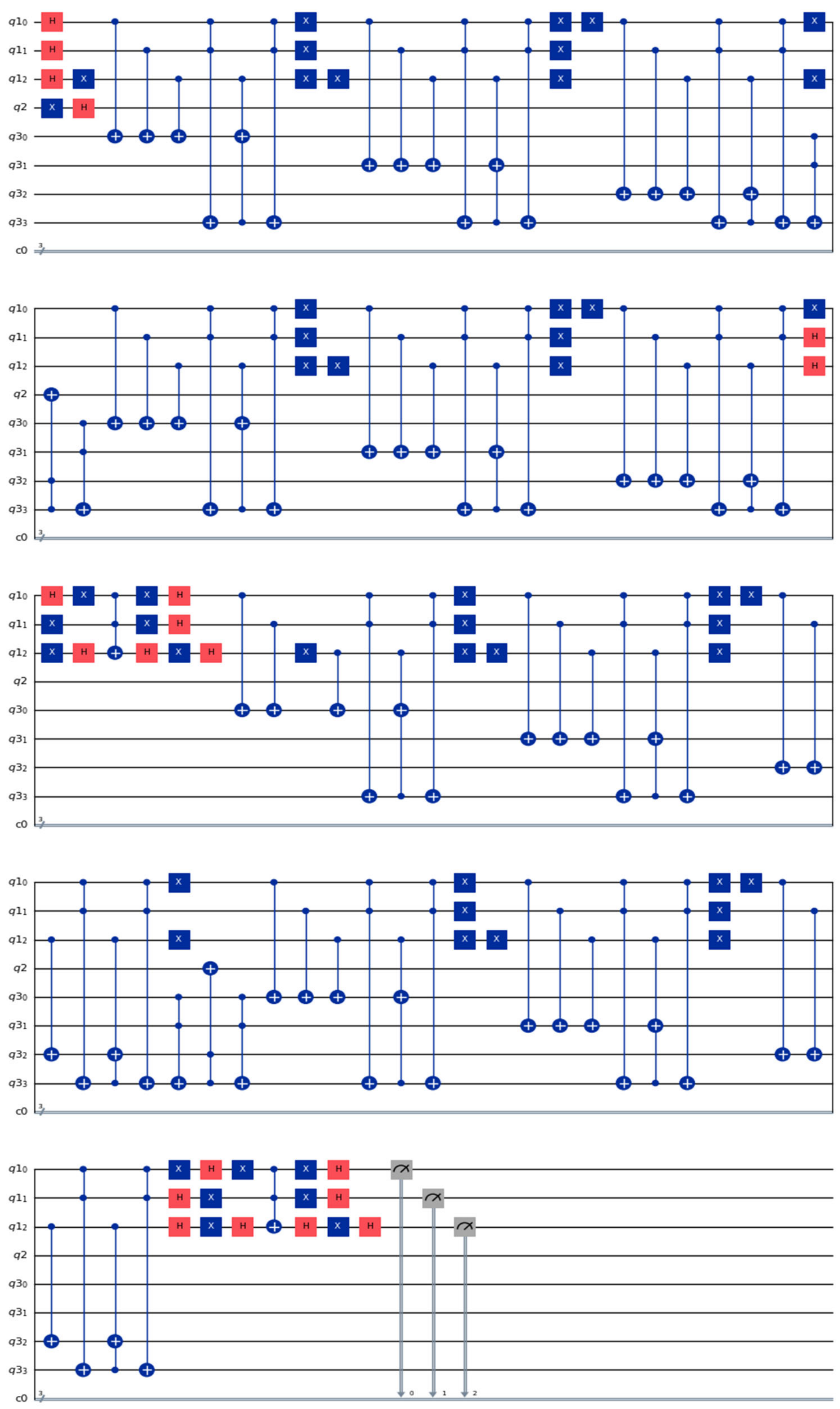

Figure 4. Implementation of a quantum circuit for Shor's algorithm in the IBM Qiskit library. In this figure, a = 7 and N = 15, and it needs a total of 8 qubits. It is drawn by the IBM Qiskit library. The blue X-box is the X-gate which is used to flip quantum state. The red H-box is the Hadamard gate. The blue dot-connected graph with + sign is the controlled X gate.

4.2. Factors Impacting the Latency Score in Quantum-Edge-Cloud

This research identifies four factors that have a direct impact on latency in the quantum-edge-cloud deployments. The transpilation optimization levels and fidelity are controllable variables, while circuit depths and the number of qubits is primarily determined by the implementations of quantum algorithms. Transpilation Optimization Levels provided by quantum computing platforms control how circuits are transformed to better fit hardware constraints. The goal is to identify the most efficient configuration that balances optimization time with computational performance. For example, on the IBM platform, Level 0 implies no optimization, Level 1 merges single-qubit gates and cancels consecutive CNOTs, Level 2 removes redundant operations, and Level 3 applies aggressive optimizations, including noise-aware qubit mapping and advanced gate cancelation [51,52]. This work discussed fidelity in Section 2.2. It provides a measure of how accurately a quantum system performs, particularly when noise affects execution reliability and speed [53–55]. Fidelity between two quantum states, represented by density matrices ρ and δ, is calculated as:

$$F(\rho, \sigma) = \mathrm{tr}\left(\sqrt{\sqrt{\delta} \quad \rho \quad \sqrt{\sigma}}\right) \tag{5}$$

In this formulation, both the states may represent mixed quantum states, which are statistical ensembles of pure states [56]. When the two states are nearly identical, the fidelity approaches 1, indicating high computational accuracy.

In quantum computing, circuit depth is the number of sequential layers of quantum gates required to implement a quantum algorithm [49]. Deeper circuits require more execution time, increasing the likelihood of errors due to qubit decoherence and gate imperfections in NISQ devices. Moreover, circuit depth affects both the transpilation process and vulnerability to noise. Optimization techniques during transpilation aim to reduce circuit depth while preserving algorithmic integrity. The number of qubits in a quantum algorithm is a fundamental determinant of system performance, as it directly influences computational capacity, error rates, and resource requirements [57]. As the number of qubits increases, the quantum system can represent exponentially larger state spaces, enabling the solution of more complex problems. However, this scalability comes with significant challenges, including susceptibility to noise [58], increased gate error rates, and longer transpilation and execution times.

4.3. The Proposed Benchmarking Framework

This section presents the mathematical formulation of the proposed benchmarking framework. The framework quantifies end-to-end performance in hybrid quantum–classical edge-cloud systems by integrating communication, transpilation, and execution latencies. Two methods are introduced to model transpilation and execution latency based on optimization levels. These levels can be represented either as ordinal values (e.g., 0–3, as defined on the IBM platform) or as normalized decimal values. These approaches are referred to as Method A (ordinal representation) and Method B (decimal representation).

4.3.1. Communication Latency

The communication latency metric is a critical component of the overall system performance, particularly in latency-sensitive applications such as real-time analytics and edge-based quantum inference. It reflects the total round trip time (RTT) required for the data to travel from the OT devices, such as IoT sensors, to the edge-cloud servers, the processing and response time at the server, and the time required for the response to return to the originating devices. The processors and links used can be classical or quantum. The formula for this calculation is expressed as follows:

$$\tau_{\mathrm{cl}} = \tau_{\mathrm{request}} + \tau_{\mathrm{setup}} + \tau_{\mathrm{response}} \tag{6}$$

where all of them are measured in seconds

τ_{request} represents transmission time from OT device to edge-cloud server.

$$\tau_{\text{request}} = \frac{|\text{request message}|}{\text{Network Speed}} \tag{7}$$

τ_{setup} represents communication processing time at the server, e.g., authentication, communication setup, message handling, etc.

τ_{response} represents the time for the response to be received back at the OT device)

$$\tau_{\text{response}} = \frac{|\text{response message}|}{\text{Network Speed}} \tag{8}$$

This metric enables precise evaluation of network-related delays and supports the identification of optimization opportunities within the edge-cloud infrastructure.

4.3.2. Transpilation Latency—Method A

Transpilation latency ($\tau_{\text{tl}}^{\text{A}}$) measures the time consumed by the transpilation procedure. It can be expressed as follows:

$$\tau_{\text{tl}}^{\text{A}} = \frac{\sum_{i=1}^{N} \tau_{\text{transpilation}}^{i} \cdot i}{N} \tag{9}$$

Here, N is the total number of optimization levels for transpilation on a specific platform, which are from quantum APIs. For instance, the IBM quantum platform offers 4 different optimization levels; thus, N = 4 on the IBM platform. $\tau_{\text{transpilation}}^{i}$ represents the transpilation latency for a specific optimization level i. The parameter i is the ordinal value of the i-th optimization level for transpilation. Though the optimization level begins with 0 in the IBM platform, its value is normally initialized to 1 in all formulas.

4.3.3. Transpilation Latency—Method B

Transpilation latency ($\tau_{\text{tl}}^{\text{B}}$) is used to measure the time consumed by the transpilation procedure, but with different weight values.

$$\tau_{\text{tl}}^{\text{B}} = \frac{\sum_{j}^{N} \tau_{\text{transpilation}}^{j} \cdot \text{weight}_j}{N} \tag{10}$$

Here, N is the total number of optimization levels, consistent with Equation (9). The term $\tau_{\text{transpilation}}^{j}$ represents the transpilation latency at optimization level j. The variable j plays a similar role to i in Equation (9), but unlike i, its value also reflects its magnitude. For instance, in Equation (9), setting i = 3 implicitly assumes a weight 1.5 times that of i = 2 across all algorithms. However, the experimental results presented in Sections 5 and 6 do not support this uniform scaling assumption. To address this, the weighted optimization levels, j, are adapting for different quantum algorithms, ensuring that the accuracy properties established in Equation (9) are preserved in Equation (11) as well. The weighting relationship is expressed as follows:

$$\text{weight}_j = \frac{\sum_{k=1}^{M} \frac{\tau_{\text{transpilation}}^{j,k}}{\tau_{\text{transpilation}}^{1,k}}}{M} \tag{11}$$

Here, j is used to express the transpilation optimization level, which may take decimal values. The weights are averaged across all quantum algorithms considered. M is the total number of algorithms used, and k indexes individual algorithms. In this study, k ∈ {1, 2, 3} since 3 quantum algorithms are selected in Section 4.5. The term $\tau_{\text{transpilation}}^{j,k}$ is the transpilation time of algorithm k at transpilation level j, while $\tau^{1,k}_{\text{transpilation}}$ is the time of operation of transpilation for no optimization level.

4.3.4. Execution Latency—Method A

Equation (12) gives the execution latency on the edge-cloud server. It uses the ordinal values i as the transpilation levels, thus $\tau^{i}_{\text{execution}}$ represents the execution time for transpilation level i.

$$\tau^{A}_{\text{execution}} = \frac{\sum_{i=1}^{N} \frac{\tau^{i}_{\text{execution}}}{\text{Fidelity}\cdot i}}{N} \tag{12}$$

4.3.5. Execution Latency—Method B

Equation (13) represents the execution latency on the edge-cloud server, incorporating the weighting factor weight_j given in Equation (14). Both Equations (12) and (13) capture the assumption that higher fidelity results in lower computation latency and, consequently, improved performance.

$$\tau^{B}_{\text{execution}} = \frac{\sum_{j=1}^{N} \frac{\tau^{j}_{\text{execution}}}{\text{Fidelity}\cdot \text{weight}_j}}{N} \tag{13}$$

$$\text{weight}_j = \frac{\sum_{k=1}^{M} \frac{\tau^{j,k}_{\text{execution}}}{\tau^{1,k}_{\text{execution}}}}{M} \tag{14}$$

Here weight_j expresses the relative execution weight corresponding to transpilation optimization level j, represented in decimal values. time rooted in the transpilation optimization level in decimal numbers. M and k follow the same definition as in Equation (11). The value $\tau^{j,k}_{\text{execution}}$ denotes the execution time of algorithm k at transpilation level j, while $\tau^{(1,k)}_{\text{execution}}$ corresponds to the execution time of algorithm k without optimization.

4.4. Experimental Environment and Computational Resources

This designed experiment employs two independent execution environments for running quantum circuits: physical QPUs and quantum simulators. Simulators are software programs that simulate QPUs on classical computers, enabling algorithm testing and validation without requiring access to quantum hardware. This study utilizes two leading quantum computing platforms, IBM Quantum Lab (Yorktown Heights, NY, USA) and Amazon Braket (Seattle, WA, USA), to conduct experiments and evaluate performance. Both platforms offer access to superconducting QPUs and simulators.

IBM Quantum Lab provides cloud-based access to IBM's proprietary quantum processors, including QPUs with up to 1000 superconducting qubits [14,58]. These QPUs are designed to perform high-fidelity quantum operations, making the platform highly suitable for executing complex quantum algorithms. IBM's performance metrics are used to assess hardware reliability and computational capabilities. The platform is supported by Qiskit (v1.2.0), an open-source Software Development Kit (SDK) that allows researchers to design, simulate, and execute quantum algorithms on both real QPUs and simulators [15].

Amazon Braket, offered by Amazon Web Services (AWS), provides a unified interface to a range of QPUs and simulators from multiple hardware providers [59]. It supports the development and deployment of quantum algorithms using Python (v3.10), allowing users to implement both pure quantum and hybrid quantum–classical workflows. Its modular architecture is well-suited for exploring advanced applications in optimization, machine learning, and cryptography.

By leveraging both platforms, this study provides a comprehensive evaluation of their capabilities, highlighting their respective strengths and suitability for different quantum computing tasks. It is further stated that to be able to meaningfully work with all the latencies, most of the experiments presume offloading to the main cloud. Both the IBM and Amazon platforms are implemented in the cloud. Users can build, test, and run quan-tum algorithms on them, leveraging both quantum and classical computing resources. Executing selected algorithms on these resources involves the communication network belonging to the telecommunications network and links owned by the cloud service provider. This allows emulation of end-to-end latencies in the hybrid edge-cloud environment. The factor of edge processing is introduced with the simplistic introduction of some preprocessing at the local server by simulating quantum processors on local CPUs.

4.5. Quantum Algorithms Selected for Latency Evaluation

To evaluate latency in hybrid quantum–classical edge-cloud environments, three representative quantum algorithms are selected: Shor's Algorithm, Grover Algorithm, and the Quantum Walks Algorithm. These algorithms represent distinct categories of quantum computational paradigms: quantum Fourier transform, amplitude amplification, and quantum walks, respectively.

- Shor's Algorithm is chosen for its ability to solve integer factorization in polynomial time $O\left((\log N)^3\right)$, which is crucial for assessing the power of quantum systems in breaking encryption.
- Grover's algorithm is selected for its efficiency in searching unsorted databases by reducing the complexity of unstructured search from $O(N)$ to $O\left(\sqrt{N}\right)$.
- The quantum walks algorithm is included to evaluate the quantum system's probabilistic modeling and simulation capabilities relevant in domains such as network routing and predictive maintenance.

These algorithms are implemented on IBM Quantum Lab and Amazon Braket to assess latency under varying system configurations and transpilation strategies.

4.5.1. Shor's Algorithm

Shor's Algorithm is a pivotal quantum algorithm that efficiently factors large integers, threatening classical encryption schemes based on the hardness of factorization [10]. The central concept of Shor's algorithm lies in identifying the period r of a specific modular arithmetic function $f(x) = a^x \bmod N$, where a is a randomly chosen integer coprime to the composite number N. The goal of the algorithm is to determine the smallest positive integer r such that:

$$a^{x+r} \equiv a^x \bmod N \tag{15}$$

Once the period r is determined using a quantum subroutine (quantum phase estimation), classical post-processing steps compute the greatest common divisor (GCD) of $a^{r/2} \pm 1$ and N, yielding the prime factors of N. This period-finding mechanism, executed on a quantum computer, demonstrates a clear quantum advantage of solving integer factorization in polynomial time $O(\log N)$ rather than classical algorithms in an infeasible amount of time for sufficiently large N. Thus, Shor's algorithm underscores the transformative potential of quantum computing in fields like cryptography, optimization, and secure communications. Figure 4 shows the implementation of Shor's algorithm on the IBM platform. Beyond cryptography, Shor's algorithm also serves as a benchmark for assessing execution depth, fidelity, and system scalability in NISQ-era devices.

4.5.2. Grover's Algorithm

Grover's algorithm is a quantum search algorithm that provides a quadratic speedup over classical approaches for searching an unsorted database or solving certain black-box function problems [11]. While classical algorithms require O(N) queries to locate a target item in a database of N elements, Grover's algorithm reduces this to $O(\sqrt{N})$, making it a foundational example of quantum algorithmic advantage. Figure 5 gives an implementation of Grover's algorithm using 3 qubits for the string "010" as contained in the IBM Qiskit library [16]. The algorithm operates on a function f(x) such that f(x) = 1 if x is the correct solution $\hat{x}$, and f(x) = 0 otherwise. Grover's method begins by creating a quantum superposition of all possible solutions. It then employs a quantum oracle to invert the amplitude of the correct solution, $\hat{x}$. Following this, an amplitude amplification process increases the probability of measuring the correct solution by systematically amplifying the marked state.

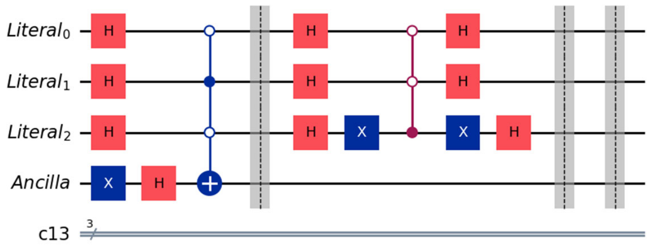

Figure 5. Implementation of Grover's algorithm with 3 qubits for string "010". The blue X-box is the X-gate, the red H-box is the Hadamard gate, and the blue dot-connected graph with + sign is the controlled X gate.

According to [60] the iterative procedure is governed by Grover's operator G, which is defined as:

$$G = (2|\psi\rangle\langle\psi| - I) * (I - 2|\hat{x}\rangle\langle\hat{x}|) \tag{16}$$

Here,

- $|\psi\rangle$ represents the equal superposition state,
- The symbol I is the identity operator.

The first term in the operator performs a reflection over the mean amplitude, while the second term applies a phase inversion to the marked state $\hat{x}$.

Repeating this operator approximately $O\left(\sqrt{N}\right)$ times amplifies the correct state's amplitude, enabling high-probability identification upon measurement.

Grover's algorithm is particularly valuable in domains such as optimization [61,62], cryptography [11], and quantum machine learning [63], where unstructured search plays a central role. Figure 5 gives an example of a quantum walk algorithm implemented in [16]. The execution procedure of Grover's algorithm is described in Figure 6. In step 1, the data is initialized. Step 2 and step 3 work together as a loop to process the amplification and oracle functions. Step 4 is the final step to return the computation result.

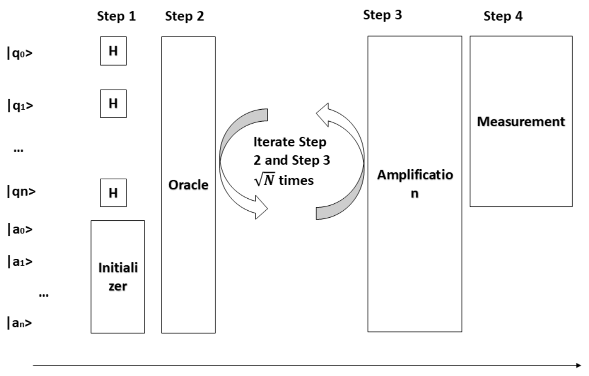

Figure 6. Execution Procedure of Grover's Algorithm. After step 1 as initialization, step 2 and step 3 repeat only $O\left(\sqrt{N}\right)$ iterations.

4.5.3. Quantum Walks Algorithm

The quantum walks algorithm is a quantum version of classical random walks that model stochastic processes in which the next state of a system is determined probabilistically by its current state [13]. It enables more efficient exploration of complex and high-dimensional structures, making them well-suited for applications in graph traversal, network analysis, and optimization. Unlike classical random walks, quantum walks leverage quantum superposition and interference, enabling more efficient exploration of complex structures. Figure 7 gives an example of the quantum walk implementation as given in the IBM Qiskit library [16]. There are two types of quantum walks: (1) discrete-time quantum walks (DTQW) and (2) continuous-time quantum walks (CTQW). DTQW does not explicitly incorporate a time parameter and is typically implemented through a sequence of unitary operations. In contrast, CTQW can be interpreted as a specialized form of quantum simulation, evolving under a time-independent Hamiltonian derived from the structure of the underlying graph [64].

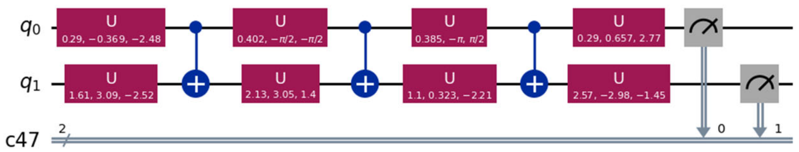

Figure 7. Implementation of the Quantum Walk Algorithm with 2 qubits.

DTQW is generally referred to as coin-based model because flipping and moving coin can be analogous to a random walk procedure. If we take I as the identity operator, C as the unitary coin operator, and the unitary shift operator S, the quantum walk can be described by a unitary as shown in Equation (17) [64].

$$\mathrm{U} = \mathrm{S}(\mathrm{C} \otimes \mathrm{I}) \tag{17}$$

In contrast, CTQW, which is originated form Markov chain [64], is governed by a differential equation involving the graph Laplacian matrix L, which encodes the connectivity of the graph. Given the graph G expressed in adjacency matrix A, the Laplacian is $\mathrm{L} = \mathrm{D} - \mathrm{A}$, where $\mathrm{D} = \mathrm{diag}(\sum_{j} \mathrm{A}_{1j}, \ldots, \sum_{j} \mathrm{A}_{nj})$ is the degree matrix [63]. Unlike DTQW, the CTWQ is defined on a Hilbert space without the need for an additional coin space [65]. Let $\mathrm{p_i(t)}$ be the probability at the position of vertex i at time t, the superposition state $|\Psi(t)\rangle$ evolution operator of the CTQW is expressed as:

$$\Big|\Psi(t)\rangle = e^{-iHt}\Big|\Psi(0)\rangle \tag{18}$$

where $\mathrm{H} = \gamma \mathrm{L}$.

The quantum walks algorithm has demonstrated significant potential in addressing network-based problems, such as finding shortest paths [66], community detection [67], and connectivity analysis. Their applications extend beyond graph theory to domains such as optimization, machine learning, and pattern recognition, where the ability to process large and complex datasets efficiently is critical. By leveraging the inherent parallelism and interference properties of quantum systems, quantum-walk algorithm provides a powerful tool for solving computationally intensive problems more effectively than their classical counterparts.

5. Experiments and Results

This section presents the experimental setup and results for measuring latency in hybrid quantum–classical systems, based on the composite latency formula defined in Equation (3). Data are collected and analyzed for the three components—communication latency, transpilation latency, and execution latency—as well as the combined Latency Score.

5.1. Communication Latency Results

Communication is common to both Method A and Method B, as both involve the transmission and reception of data across the same network configuration. To simulate real-world conditions, a VirtualBox-based testbed is deployed. Two virtual instances are synchronized using identical system clocks, and Python scripts are used to emulate message exchanges between an edge device and a quantum service provider. Using a baseline network bandwidth of 56 kbps, the round-trip communication time is measured based on Equations (7) and (8). While 56 kbps is chosen for simplicity, this parameter can be adjusted for models in different network scenarios.

5.1.1. Communication Latency: A Case Study with Grover's Algorithm

Grover's algorithm can be adapted to a wide range of search problems. In our experiment, to evaluate communication latency in practice, Grover's algorithm is implemented using an oracle designed to identify the binary value "110", represented using 3 qubits. The request message is structured as {name: 'grover', n: 110}.

This message is 27 bytes in size and is sent from the edge device to the quantum server.

On the IBM Quantum platform:

- Simulators return output as a histogram of quantum states and their occurrence counts.
- QPUs return measured results in a similar format, though with more variation due to noise.

  For example:

- Simulator Output (Optimization Level 1):

  {'110': 95, '000': 3, '101': 2}
  Indicates a strong peak for the target state.

- QPU Output (Optimization Level 1):

  {'110': 39, '100': 14, '111': 13, '011': 8, '001': 6, '101': 6, '000': 8, '010': 6}
  Reflects noise influence on measurement outcomes.

On the Amazon Braket platform:

- The output is a probability distribution over all 8 possible 3-qubit states:

  [0.03, 0.04, 0.06, 0.02, 0.03, 0.03, 0.76, 0.03]

This corresponds to states ['000', '001', '010', '011', '100', '101', '110', '111'], with state '110' having the highest probability (0.76), as expected.

The total size of response messages, including measurement results and metadata, is recorded in Tables 1 and 2, allowing for accurate calculation of the communication latency component.

This case study illustrates the variability in message formats and data volume across platforms and execution environments, which must be accounted for when benchmarking latency in hybrid quantum–classical systems.

Table 1. Response messages on the IBM platform for Grover's Algorithm.

Optimization Levels	On Simulator	On QPUs
1	['110': 95, '000': 3, '101': 2]	['110': 39, '100': 14, '111': 13, '011': 8, '001': 6, '101': 6, '000': 8, '010': 6]
...	...	...
Message Length in Bytes	222	340

Table 2. Response messages on the Amazon Braket platform for Grover's Algorithm.

Platform	On Simulator	On QPUs
Amazon Braket	[0.03 0.04 0.06 0.02 0.03 0.03 0.76 0.03]	[0.02 0.1 0.09 0.08 0.06 0.02 0.53 0.1]
Message Length in Bytes	48	46

5.1.2. Communication Latency: A Case Study with Shor's Algorithm

In the experimental implementation of Shor's algorithm, parameters were set at a = 7 and N = 15 consistent with Equation (15). These values configure the quantum circuit to perform integer factorization by identifying the period r of the function:

$$f(x) = a^x \bmod N$$

This configuration requires a total of 8 qubits for the quantum circuit to represent the necessary quantum state space.

The request message for this computation is encoded in the JSON format as {name: 'shors', a: 7, N: 15}.

This yields a message size of 39 bytes. Once the computation is complete, the system returns measurement results in platform-specific formats.

Table 3 gives the response from IBM's simulator at optimization level 1, with measurement outcomes such as

['10000000': 27, '0000000': 27, '01000000': 21, '11000000': 25]

Each 8-bit binary string represents a quantum state, and the associated value indicates the number of times that state was observed over multiple runs. For example, '10000000' has an occurrence of 27 and '01000000' has an occurrence of 21.

Table 4 shows the same formatted response messages as returned from the Amazon platform.

Table 3. Response message on IBM platform for Shor's Algorithm.

Optimization Levels	On Simulator	On QPUs
Level 1	['10000000': 27, '0000000': 27, '01000000': 21, '11000000': 25]	['10000100': 1, '01101111': 1, '10100010': 2, ... '01001110': 2]
...	...	...
Total Message Length in Bytes	264	558

These response messages serve as the basis for calculating the communication latency component τ_{cl} by measuring the size and transmission time of both the request and the response data. This case highlights the impact of circuit complexity and message formatting on communication latency in hybrid quantum-cloud systems. Shor's algorithm, due to its higher qubit count and result complexity, typically produces larger message payloads than Grover's algorithm, making it an important factor in evaluating edge-cloud communication overhead.

Table 4. Response message on the Amazon Braket platform for Shor's Algorithm.

Platform	On Simulator	On QPUs
Amazon Braket	['00001110': 10, '00000010': 8, . . ., '11001011': 2]	['11111111': 20, '11110111': 7, . . . '11111010': 1]
Total Message Length in Bytes	256	828

5.1.3. Communication Latency: Case Study with Quantum Walks Algorithm

In this experiment, the Quantum Walks algorithm is implemented on a 4-nodes circular graph to investigate its probabilistic behavior across different numbers of steps. evaluate its dynamic behavior over varying numbers of steps. The analysis focuses on the occurrence counts of quantum states after 1-step, 2-step, 3-step, and 4-step transitions, which are crucial for understanding the probabilistic nature of the algorithm.

The request message initiating the execution is encoded in the JSON format as {name: ′quantumwalk′, n: 4, step: 4},

This results in a total data volume of 42 bytes. The experiment evaluates the quantum walks on both the IBM Quantum and Amazon Braket platforms.

Tables 5–7 present the results. Take Table 6, for instance; the response message is

['10': 4, '11': 2, '01': 10, '00': 84]

These binary strings represent the node positions in the quantum walks, while the associated values indicate their occurrence count. The states '10', '11', '01', and '00' are used to mark steps in the quantum walk algorithms.

Table 5. Response messages on the IBM simulator for the Quantum Walks Algorithm.

	Optimization Levels			
Quantum Walks Steps	**1**	**2**	**3**	**4**
1	['00': 100]	['00': 100]	['00': 100]	['00': 100]
. . .	. . .	. . .	. . .	. . .
Total Message Length in Bytes	113	122	124	124

Table 6. Response messages on IBM QPU for Quantum Walks Algorithm.

	Optimization Levels			
Steps	**1**	**2**	**3**	**4**
1	['10': 4, '11': 2, '01': 10, '00': 84]	['11': 1, '01': 6, '10': 6, '00': 87]	['00': 100]	['00': 100]
. . .	. . .	. . .	. . .	. . .
Total Message Length in Bytes	52	102	102	162

Table 7. The response message for the Quantum Walks Algorithm on the Amazon Braket platform.

Algorithms	On Simulator	On QPUs
Quantum Walks	[1: 0.512, 3: 0.488], [0: 0.54, 2: 0.463], [3: 1.0], [2: 1.0]	[1: 0.512, 3: 0.488], [0: 0.537, 2: 0.463], [3: 1.0], [2: 1.0]
Total Message Length in Bytes	90	90

Table 7 provides results in terms of probability distributions over positions. For instance:

[1: 0.512, 3: 0.488]

Indicates a 51.2% probability of the walker being at node 1 and 48.8% at node 3 after a given step, highlighting the algorithm's inherent quantum interference.

The combination of these response formats across platforms enables a detailed evaluation of communication latency, particularly in relation to the complexity of the quantum state and output size.

This case study emphasizes how the quantum walks algorithm—especially in multi-step executions—can produce varying output data volumes, influencing the latency profile. The experimental setup demonstrates the importance of considering algorithm-specific output characteristics when benchmarking hybrid quantum-cloud systems.

A comparative summary of the communication latency case studies is given in Table 8.

Table 8. Comparative Summary of Communication Latency Case Studies.

Feature	Grover's Algorithm	Shor's Algorithm	Quantum Walks Algorithm
Algorithm Type	Amplitude amplification	Period finding/integer factorization	Probabilistic traversal/graph search
Use Case	Unstructured search	Cryptographic analysis	Graph-based modeling, optimization
Platform(s) Used	IBM Quantum, Amazon Braket	IBM Quantum, Amazon Braket	IBM Quantum, Amazon Braket
Request Message Format	{name: 'grover', *n*: 110}	{name: 'shors', a: 7, N: 15}	{name: 'quantumwalk', *n*: 4, step: 4}
Request Message Size	27 bytes	39 bytes	42 bytes
Number of Qubits Used	3 qubits	8 qubits	2–3 qubits (for a 4-node graph)
Response Format (IBM)	Frequency counts of quantum states (e.g., {'110': 95, '000': 3})	Frequency counts of 8-bit states (e.g., {'10000000': 27})	Frequency counts of 2-bit states (e.g., {'00': 84, '01': 10})
Response Format (Amazon)	Probability vector over 8 states (e.g., [0.76 for '110'])	Similar frequency-based results	Node-indexed probability map (e.g., {1: 0.512, 3: 0.488})
Output Data Size	Small to medium	Medium to large	Small to medium
Latency Sensitivity	Low to moderate	High (larger messages, more qubits)	Moderate (repeats per step)
Communication Complexity Impact	Minimal message variance	High due to circuit/output size	Varies with number of steps

5.2. *Transpilation Latency*

To make the result comparable, its transpilation time is recorded by Python rather than relying on the timestamp provided by the platform servers. This approach allows precise control over data collection across experiments.

It is important to note that Amazon Braket performs transpilation as a black-box process, offering no access to or configuration of transpilation parameters. Consequently, for analysis on Braket, transpilation latency is recorded as zero, reflecting the absence of tunable optimization settings.

Due to the current lack of availability of quantum processors at the customized edge cloud, all experiments have been conducted on the main cloud infrastructure provided by IBM and Amazon. Nevertheless, because the tests are performed using both simulators and real QPUs on their respective native platforms, the results yield a valid and insightful basis for comparative performance evaluation.

5.2.1. Transpilation Latency of Grover's Algorithm

Table 9 records the transpilation latency measured in seconds for Grover's algorithm on the IBM simulator and QPUs at different optimization levels.

Table 9. Transpilation time (in seconds) on the IBM platform with different optimization levels for Grover's Algorithm.

	Optimization Levels			
	Level 1	**Level 2**	**Level 3**	**Level 4**
Simulator	2.3×10^{-2}	1.5×10^{-2}	4.7×10^{-2}	2.0×10^{-2}
QPUs	1.29×10^{-1}	2.1×10^{-2}	2.2×10^{-2}	2.1×10^{-2}

5.2.2. Transpilation Latency of Shor's Algorithm

Table 10 records the transpilation time in seconds for Shor's algorithm on the IBM simulator and QPUs at different optimization levels.

Table 10. Transpilation time (in seconds) on IBM with different optimization levels for Shor's Algorithm.

	Optimization Levels			
Transpilation Time	**Level 1**	**Level 2**	**Level 3**	**Level 4**
On Simulator	1.25	2.45×10^{-1}	1.06	1.52
On QPUs	2.49×10	2.47×10	8.28×10	1.24×10^{2}

5.2.3. Transpilation Latency of Quantum Walks Algorithm

Tables 11 and 12 show the transpilation time in seconds for different steps on IBM's platform.

Table 11. Transpilation time (in seconds) on IBM's Simulator for Quantum Walks Algorithm.

	Optimization Levels			
Steps	**Level 1**	**Level 2**	**Level 3**	**Level 4**
1	1.02	6.0×10^{-2}	6.3×10^{-2}	6.6×10^{-2}
2	8.9×10^{-2}	9.8×10^{-2}	1.09×10^{-1}	1.22×10^{-1}
3	9.6×10^{-2}	9.5×10^{-2}	1.12×10^{-1}	1.08×10^{-1}
4	1.18×10^{-1}	9.9×10^{-2}	1.11×10^{-1}	1.1×10^{-1}

Table 12. Transpilation time (in seconds) on IBM's QPU for the Quantum Walks Algorithm.

	Optimization Levels			
Steps	**1**	**2**	**3**	**4**
1	5.0×10^{-2}	2.7×10^{-2}	3.4×10^{-2}	4.3×10^{-2}
2	6.3×10^{-2}	6.3×10^{-2}	1.3×10^{-1}	1.17×10^{-1}
3	1.97×10^{-1}	6.3×10^{-2}	8.5×10^{-2}	1.1×10^{-1}
4	8.2×10^{-2}	1.19×10^{-1}	2.87×10^{-1}	1.22×10^{-1}

5.3. Execution Latency

The execution phase is evaluated using two simulation environments: the qasm_simulator on IBM Quantum and the local simulator on Amazon Braket. Each experiment is conducted with 100 shots to ensure statistical significance, which explains why the total occurrences in the result tables (Section 5.2) consistently sum to 100.

Due to limitations in current APIs, the real-time fidelity could not be captured directly during execution. Instead, fidelity values are based on reported benchmarks from prior studies. Specifically, IBM's QPUs are assumed to operate at an average fidelity of 94.98% [65], while Amazon Braket reports a fidelity of 99.92% [68]. In a simulated environment, fidelity is assumed to be 100% under the assumption of a noise-free system [1], which enables a clearer evaluation of computational performance without noise-caused errors.

Following this experimental setup, the execution latency is measured, and the three selected quantum algorithms are computed and compared across both platforms. A detailed analysis of the results for these algorithms is provided in the following subsections, offering insight into platform-specific performance and the computational demands of each algorithm.

5.3.1. Execution Latency of Grover's Algorithm

Table 13 records the execution time on IBM's simulator and QPUs. Table 14 records the execution time on the Amazon simulator and QPUs.

Table 13. Execution time (in seconds) on the IBM platform with different optimization levels for Grover's Algorithm.

	Optimization Levels			
Execution Time	**Level 1**	**Level 2**	**Level 3**	**Level 4**
Simulator	8.7×10^{-2}	6.4×10^{-2}	1.8×10^{-2}	3.2×10^{-2}
QPUs	2.37×10	3.66×10	2.98×10	2.52×10

Table 14. Execution time (in seconds) on Amazon for Grover's Algorithm.

Amazon Platform	**Execution time**
Simulator	4.86×10^{-5}
QPUs	1.23×10^{2}

5.3.2. Execution Latency of Shor's Algorithm

Tables 15 and 16 further illustrate Shor's algorithm's execution time in seconds on IBM and Amazon's platforms, respectively.

Table 15. Execution time (in seconds) on the IBM platform with different optimization levels for Shor's Algorithm.

	Optimization Levels			
Execution Time	**Level 1**	**Level 2**	**Level 3**	**Level 4**
Simulator	1.48×10^{-1}	1.89×10^{-1}	2.37×10^{-1}	1.77×10^{-1}
QPUs	9.45×10	6.17×10	7.41×10	8.09×10

Table 16. Execution time (in seconds) on the Amazon platform for Shor's Algorithm.

Platform	**Execution Time**
Simulator	3.30×10^{-1}
QPUs	1.32×10

5.3.3. Execution Latency of Quantum Walks Algorithm

Tables 17–19 show the execution time in seconds of different steps on IBM's and Amazon's platforms, respectively.

Table 17. Execution time (in seconds) on IBM's Simulator for Quantum Walks Algorithm.

Execution Time	Optimization Levels			
Steps	**Level 1**	**Level 2**	**Level 3**	**Level 4**
1	3.1×10^{-3}	4.0×10^{-3}	1.7×10^{-3}	1.8×10^{-3}
2	2.87×10^{-2}	2.1×10^{-3}	4.5×10^{-3}	2.7×10^{-3}
3	2.5×10^{-3}	2.1×10^{-3}	2.5×10^{-3}	1.29×10^{-2}
4	2.1×10^{-3}	2.29×10^{-2}	7.8×10^{-3}	3.0×10^{-3}

Table 18. Execution time (in seconds) on IBM's QPU for the Quantum Walks Algorithm.

Execution Time	Optimization Levels			
Steps	**1**	**2**	**3**	**4**
1	3.07×10	3.05×10	2.59×10	3.05×10
2	3.28×10	3.65×10	3.08×10	3.09×10
3	3.00×10	3.69×10	3.00×10	2.46×10
4	2.56×10	3.83×10	3.07×10	3.67×10

Table 19. Execution time (in seconds) on Amazon for the Quantum Walks Algorithm.

Platform	Execution Time
Simulator	3.18×10^{-1}
QPUs	1.56×10

5.4. *The Latency Score—Method A*

The Latency Score is calculated based on Formula (3). Table 20 presents the latency scores calculated on the IBM Quantum platform and Amazon Quantum Platform, encompassing both simulators and QPUs.

Table 20. Method A—Latency Score of IBM Platform and Amazon Platform (in seconds).

Algorithms	IBM Simulator	IBM QPU	Amazon Simulator	Amazon QPU
Grover's	1.37×10^{-1}	1.47×10	1.08×10^{-2}	1.23×10^{2}
Shor's	2.89	2.47×10^{2}	3.71×10^{-1}	1.34×10
Quantum Walks	3.81×10^{-1}	1.66×10	3.37×10^{-1}	1.56×10

5.5. *Evaluation of Method A*

Until now, the time data used for transpilation and execution have been recorded and plotted in Figures 8–11 below. It is clear that their trends are not consistent as a whole, meaning the ordinal values (in images, they are marked as i = {1, 2, 3, 4}) used in Formulas (9) and (12) might lead to bias in explaining data trends. Thus, the Method B model is proposed to adjust the j values first, then update the latency values in Formulas (10) and (13).

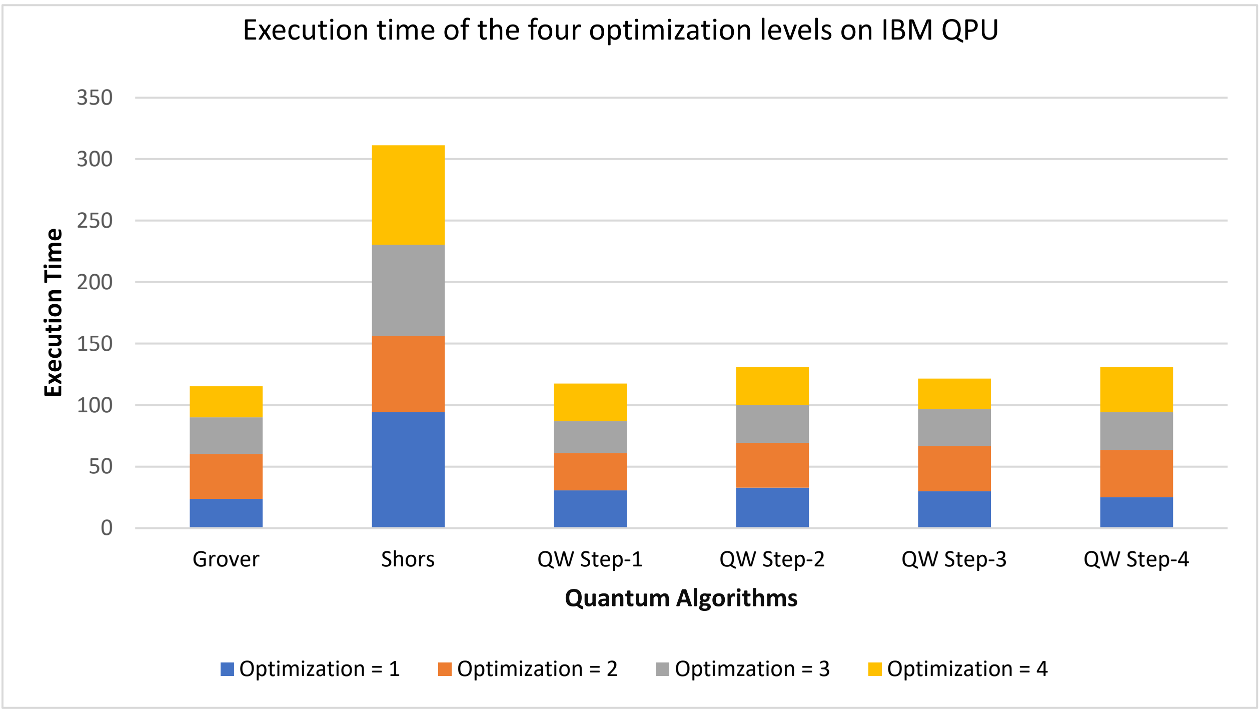

Figure 8. Execution time of different optimization levels on IBM's QPU. Shor's algorithm dominates other algorithms.

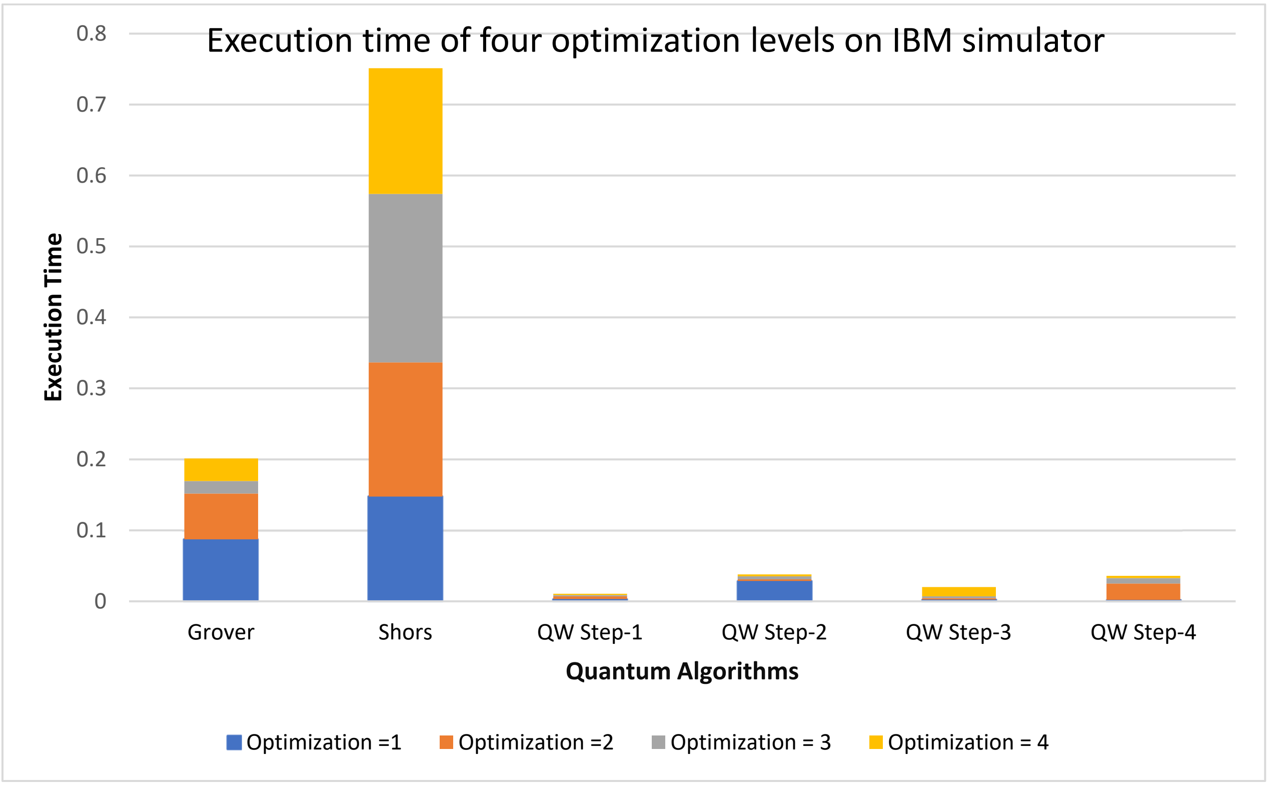

Figure 9. Execution time of different optimization levels on IBM's simulator. Shor's algorithm dominates other algorithms.

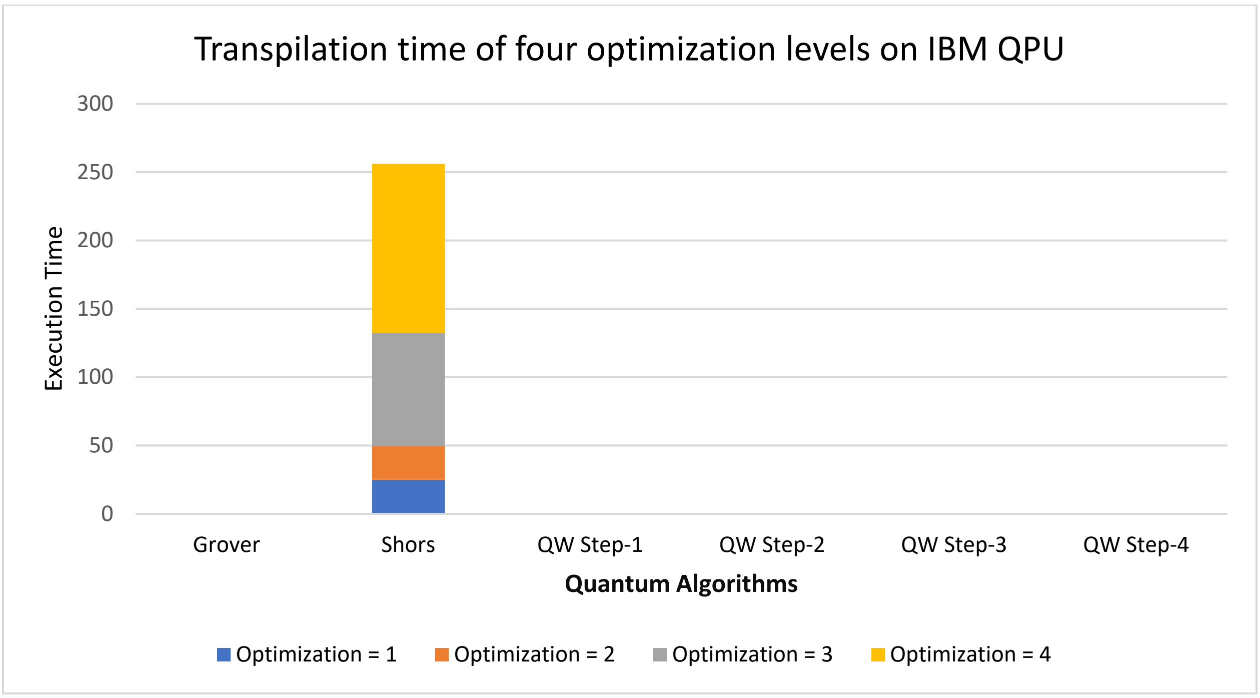

Figure 10. Transpilation time of different optimization levels on IBM's QPU. Shor's algorithm dominates other algorithms. The others have very low values.

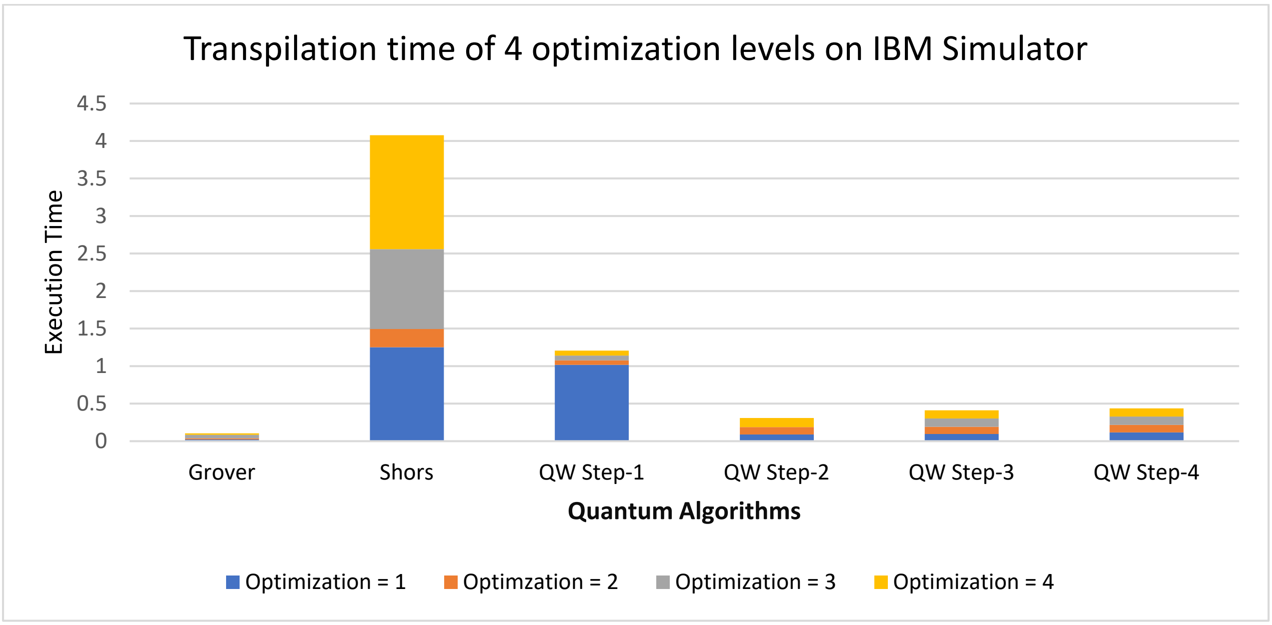

Figure 11. Transpilation time of different optimization levels on IBM's simulator. Shor's algorithm dominates other algorithms.

5.6. The Latency Score—Method B

The Latency Score is calculated based on Formula (3). Table 21 presents the latency scores calculated on the IBM Quantum platform and Amazon Quantum Platform, encompassing both simulators and QPUs.

Table 21. Method B—Latency Score of the IBM platform and Amazon Platform (in seconds).

Algorithms	IBM Simulator	IBM QPU	Amazon Simulator	Amazon QPU
Grover's	1.59×10^{-1}	2.38×10	1.08×10^{-2}	1.23×10^{2}
Shor's	1.20	3.29×10^{2}	3.71×10^{-1}	1.34×10
Quantum Walks	1.32×10^{-1}	2.71×10	3.37×10^{-1}	1.56×10

6. Analysis and Discussion of the Experimental Results

6.1. Analysis of Experimental Results

This section presents a detailed analysis of the experimental results, emphasizing their implications for quantum computing and hybrid edge-cloud integration.

1. Impact of Optimization Levels on Execution Time

IBM Quantum Lab offers compiler optimization levels ranging from 0 to 3 (denoted as optimization = {1, 2, 3, 4} in Figures 8–11). These levels can improve execution performance on both simulators and QPUs. As shown in Figures 8 and 9, execution time varies significantly across algorithms and the optimization levels. Figures 8 and 9 illustrate the execution times under varying optimization levels on IBM platforms. These findings suggest that the effectiveness of compiler optimizations is algorithm- and task-dependent.

2. Comparative Performance of Simulators and QPUs

Experimental results reveal that simulators on both IBM and Amazon platforms consistently outperform their respective QPUs, as reflected by lower latency scores in both Method A and Method B (Tables 20 and 21). Current QPUs restrict researchers from using a large number of qubits for experimentation, preventing full utilization of quantum advantage. Availability of a limited number of qubits restricts the complexity of implementable algorithms. Consistent with existing literature, QPUs capable of addressing complex real-world problems—such as secure hybrid edge-cloud deployments—are still estimated to be at least five years away from practical viability.

3. Transpilation Time Analysis

Transpilation time measures the duration required to convert a logical quantum circuit into a hardware-specific implementation. This metric is strongly influenced by the number of qubits. Figures 10 and 11 show transpilation times for different optimization levels on IBM platforms. Among the tested algorithms, Shor's algorithm demonstrates the longest transpilation and execution times on both the simulator and QPU, as it utilizes eight qubits—the highest number among the three algorithms. This underscores the computational overhead associated with larger quantum circuits and highlights the need for optimization strategies to reduce transpilation time.

4. Impact of Fidelity

While IBM's optimization levels (0 to 3) yield only marginal improvements in fidelity, they cause substantial variations in execution time and computational efficiency. This reinforces the role of compiler optimization in determining the overall system performance. A refined benchmarking approach that accounts for different optimization levels could yield more precise assessments of their effects on quantum computation efficiency. These findings point to the need for a more nuanced benchmarking approach that incorporates optimization level as a weighted factor in assessing system performance and algorithmic efficiency.

5. Overall Platform Performance

Although IBM QPUs feature a higher nominal qubit count, Amazon's platform demonstrates superior performance in two of the three evaluated algorithms (Tables 20 and 21). This discrepancy likely stems from architectural differences, optimization strategies, and algorithm-specific characteristics. These findings confirm that qubit count alone is not a sufficient indicator of platform performance. Instead, comprehensive evaluation should include optimization techniques, error mitigation methods, and overall system integration to accurately reflect performance.

6.2. Conclusions

This research investigates the integration of quantum computing technologies into classical edge-cloud networks, leveraging publicly available platforms such as IBM Quantum Lab and Amazon Braket. By implementing well-known quantum algorithms and using latency scores as the primary performance metric, this research conducted a comparative evaluation of simulators and QPUs across varying optimization levels and system configurations. The key conclusions are as follows:

(a) In all experimental scenarios, quantum simulators demonstrated superior latency performance compared to physical quantum processing units. This outcome underscores the current limitations in QPU technological maturity, particularly with respect to scalability, reliability, and noise resilience.

(b) Although IBM offers a higher qubit count, Amazon's platform achieved better performance for all tested algorithms. This finding highlights that system architecture, compiler efficiency, and optimization methodologies exert greater influence than raw hardware specifications. Thus, qubit count alone is an insufficient measure of platform capability.

(c) The experiments reveal that transpilation introduces considerable overhead, which can adversely affect total execution time. This emphasizes the necessity of striking a balance between optimization strategies during transpilation and practical constraints of execution.

7. Future Research

This study demonstrates the transformative potential of quantum computing in edge-cloud environments and establishes a foundational framework for future investigations. As discussed, most of the current experiments assume the availability of quantum–classical resources in the cloud-based platforms. This is consistent with the availability of quantum platforms like Amazon and IBM on their respective clouds. To emulate edge processing, limited pre-processing is performed locally by simulating quantum processors on classical CPUs, thereby enabling execution of selected algorithms and measurement of end-to-end latencies. Moving forward, our research will expand toward more complex distributed environments with hybrid processing taking place at all the layers.

Several promising research directions are envisioned.

(1) A deeper investigation of the interaction between transpilation optimization levels and QPU topology will help clarify the trade-offs between aggressive optimization and execution overhead.

(2) Incorporating a wider variety of quantum algorithms will refine benchmarking formulas, update key parameters, and generate broader insights into platform behavior across diverse computational tasks.

(3) Advances in practical QPUs are expected to enable flexible allocation of qubits, allowing tasks to leverage computational power dynamically according to problem complexity.

(4) Extending the framework from classical to quantum communication networks may provide more accurate insights into true end-to-end performance in hybrid quantum–edge-cloud systems.
(5) Incorporating real-time fidelity monitoring, rather than static fidelity values, will improve the accuracy of performance assessments and strengthen benchmarking methodologies.

Collectively, these directions aim to advance the practical integration of quantum computing into next-generation edge-cloud architectures, bridging the gap between current experimental prototypes and scalable, real-world deployments.

Author Contributions: Conceptualization, L.G.; Methodology, G.Y. and LG.; Software, G.Y.; Investigation, G.Y. and LG.; Data curation, G.Y.; Writing—original draft, G.Y.; Writing—review & editing, G.Y. and L.G.; Visualization, G.Y. and LG.; Supervision, L.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Resch, S.; Karpuzcu, U.R. Benchmarking quantum computers and the impact of quantum noise. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 142. [CrossRef]
2. Grover, L.K. Synthesis of quantum superpositions by quantum computation. *Phys. Rev. Lett.* **2000**, *85*, 1334. [CrossRef]
3. What Is Quantum Computing? Available online: https://scienceexchange.caltech.edu/topics/quantum-science-explained/quantum-computing-computers (accessed on 17 February 2025).
4. Gupta, M.; Nene, M.J. Quantum computing: An entanglement measurement. In Proceedings of the 2020 IEEE International Conference on Advent Trends in Multidisciplinary Research and Innovation (ICATMRI), Buldhana, India, 30 December 2020; pp. 1–6. [CrossRef]
5. What Are the Applications of Quantum Entanglement? Consensus. Available online: https://consensus.app/questions/what-applications-quantum-entanglement/ (accessed on 17 July 2025).
6. Hossain, M.I.; Sumon, S.A.; Hasan, H.M.; Akter, F.; Badhon, M.B.; Islam, M.N.U. Quantum-Edge Cloud Computing: A Future Paradigm for IoT Applications. *arXiv* **2024**, arXiv:2405.04824. [CrossRef]
7. Acuaviva, A.; Aguirre, D.; Peña, R.; Sanz, M. Benchmarking Quantum Computers: Towards a Standard Performance Evaluation Approach. *arXiv* **2024**, arXiv:2407.10941. [CrossRef]
8. Varghese, B.; Wang, N.; Bermbach, D.; Hong, C.H.; Lara, E.D.; Shi, W.; Stewart, C. A survey on edge performance benchmarking. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–33. [CrossRef]
9. Hao, T.; Hwang, K.; Zhan, J.; Li, Y.; Cao, Y. Scenario-based AI benchmark evaluation of distributed cloud/edge computing systems. *IEEE Trans. Comput.* **2023**, *72*, 719–731. [CrossRef]
10. Lorenz, J.M.; Monz, T.; Eisert, J.; Reitzner, D.; Schopfer, F.; Barbaresco, F.; Kurowski, K.; Schoot, W.; Strohm, T.; Senellart, J.; et al. Systematic benchmarking of quantum computers: Status and recommendations. *arXiv* **2025**, arXiv:2503.04905. [CrossRef]
11. Cross, A.W.; Bishop, L.S.; Sheldon, S.; Nation, P.D.; Gambetta, J.M. Validating quantum computers using randomized model circuits. *Phys. Rev. A* **2019**, *100*, 032328. [CrossRef]
12. Shor, P.W. Algorithms for quantum computation: Discrete logarithms and factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134. [CrossRef]
13. Grover, L.K. A fast quantum mechanical algorithm for database search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, USA, 22–24 May 1996; pp. 212–219. [CrossRef]
14. Shenvi, N.; Kempe, J.; Whaley, K.B. Quantum random-walk search algorithm. *Phys. Rev. A* **2023**, *67*, 052307. [CrossRef]
15. Q-CTRL's Performance Management on 127-Qubit IBM Brisbane Processor. Available online: https://quantumzeitgeist.com/q-ctrls-performance-management-on-127-qubit-ibm-brisbane-processor/ (accessed on 13 September 2024).
16. Wille, R.; Van Meter, R.; Naveh, Y. IBM's Qiskit tool chain: Working with and developing for real quantum computers. In Proceedings of the 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 25–29 March 2019; pp. 1234–1240. [CrossRef]

17. Aerts, D. Quantum Mechanics: Structures, Axioms and Paradoxes. In *Quantum Structures and the Nature of Reality: The Indigo Book of "Einstein Meets Magritte"*; Springer: Dordrecht, The Netherlands, 1999. [CrossRef]
18. Rieffel, E.; Wolfgang, P. An introduction to quantum computing for non-physicists. *ACM Comput. Surv. (CSUR)* **2020**, *32*, 300–335. [CrossRef]
19. Dirac, P.A.M. *The Principles of Quantum Mechanics*; No. 27; Oxford University Press: Oxford, UK, 1981.
20. Berberich, J.; Daniel, F. Quantum computing through the lens of control: A tutorial introduction. *IEEE Control. Syst.* **2024**, *44*, 24–49. [CrossRef]
21. Dirac, P.A.M. A new notation for quantum mechanics. In *Mathematical Proceedings of the Cambridge Philosophical Society*; Cambridge University Press: Cambridge, UK, 1939; Volume 35, pp. 416–418. [CrossRef]
22. Furutanpey, A. Architectural Vision for Quantum Computing in the Edge-Cloud Continuum. In Proceedings of the 2023 IEEE International Conference on Quantum Software (QSW), Chicago, IL, USA, 2–8 July 2023. [CrossRef]
23. Rambo, T.M.; McCusker, K.; Huang, Y.P.; Kumar, P. Low-loss all-optical quantum switching. In Proceedings of the 2013 IEEE Photonics Society Summer Topical Meeting Series, Waikoloa, HI, USA, 8–10 July 2013; pp. 179–180. [CrossRef]
24. Aliro. What are Quantum Repeaters? Available online: https://www.aliroquantum.com/blog/what-are-quantum-repeaters (accessed on 17 July 2025).
25. Collins, D.; Nicolas, G.; Hugues, D.R. Quantum relays for long distance quantum cryptography. *J. Mod. Opt.* **2005**, *52*, 735–753. [CrossRef]
26. Awschalom, D.; Berggren, K.K.; Bernien, H.; Bhave, S.; Carr, L.D.; Davids, P.; Economou, S.E.; Englund, D.; Faraon, A.; Fejer, M.; et al. Development of Quantum Interconnects (QuICs) for Next-Generation Information Technologies. *PRX Quantum* **2021**, *2*, 017002. [CrossRef]
27. Swayn, M. Quantum Companies Battle for SPAM Dominance. Quantum Insider. 2024. Available online: https://thequantuminsider.com/2022/03/04/quantum-companies-battle-for-spam-dominance/ (accessed on 18 June 2025).
28. Sim, S.E.; Steve, E.; Richard, R.H. Using benchmarking to advance research: A challenge to software engineering. In Proceedings of the 25th International Conference on Software Engineering, Portland, OR, USA, 3–10 May 2003; pp. 74–83. [CrossRef]
29. Weber, L.M.; Saelens, W.; Cannoodt, R.; Soneson, C.; Hapfelmeier, A.; Gardner, P.P.; Boulesteix, A.L.; Saeys, Y.; Robinson, M.D. Essential guidelines for computational method benchmarking. *Genome Biol.* **2019**, *20*, 125. [CrossRef] [PubMed]
30. Proctor, T.; Young, K.; Baczewski, A.D.; Blume-Kohout, R. Benchmarking quantum computers. *Nat. Rev. Phys.* **2025**, *7*, 105–118. [CrossRef]
31. Quetschlich, N.; Burgholzer, L.; Wille, R. MQT Bench: Benchmarking software and design automation tools for quantum computing. *Quantum* **2023**, *7*, 1062. [CrossRef]
32. Nation, P.D.; Saki, A.A.; Brandhofer, S.; Bello, L.; Garion, S.; Treinish, M.; Javadi-Abhari, A. Benchmarking the performance of quantum computing software for quantum circuit creation, manipulation and compilation. *Nat. Comput. Sci.* **2025**, *5*, 427–435. [CrossRef]
33. Mitchem, S. What is Quantum Coherence. Argonne National Laboratory, 19 February 2025. Available online: https://www.anl.gov/article/what-is-quantum-coherence (accessed on 10 June 2025).
34. Wudarski, F.; Marshall, J.; Petukhov, A.; Rieffel, E. Augmented fidelities for single-qubit gates. *Phys. Rev. A* **2020**, *102*, 052612. [CrossRef]
35. Vadali, A.; Kshirsagar, R.; Shyamsundar, P.; Perdue, G.N. Quantum circuit fidelity estimation using machine learning. *Quantum Mach. Intell.* **2024**, *6*, 1. [CrossRef]
36. Microsoft. Explore Quantum. Available online: https://quantum.microsoft.com/en-us/insights/education/concepts/quantum-error-correction (accessed on 18 June 2025).
37. Michielsen, K.; Nocon, M.; Willsch, D.; Jin, F.; Lippert, T.; De Raedt, H. Benchmarking gate-based quantum computers. *Comput. Phys. Commun.* **2017**, *220*, 44–55. [CrossRef]
38. Weder, B.; Barzen, J.; Leymann, F.; Zimmermann, M. Hybrid quantum applications need two orchestrations in superposition: A software architecture perspective. In Proceedings of the 2021 IEEE International Conference on Web Services (ICWS), Chicago, IL, USA, 5–10 September 2021; pp. 1–13.
39. Gupta, L. Collaborative Edge-Cloud AI for IoT Driven Secure Healthcare System. In Proceedings of the 2023 IEEE International Systems Conference (SysCon), Vancouver, BC, Canada, 17–20 April 2023; pp. 1–8. [CrossRef]
40. Tomesh, T.; Gokhale, P.; Omole, V.; Ravi, G.S.; Smith, K.N.; Viszlai, J.; Wu, X.-C.; Hardavellas, N.; Martonosi, M.R.; Chong, F.T. Supermarq: A scalable quantum benchmark suite. In Proceedings of the 2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA), Seoul, Republic of Korea, 2–6 April 2022; pp. 587–603. [CrossRef]
41. Lubinski, T.; Johri, S.; Varosy, P.; Coleman, J.; Zhao, L.; Necaise, J.; Baldwin, C.H.; Mayer, K.; Proctor, T. Application-oriented performance benchmarks for quantum computing. *IEEE Trans. Quantum Eng.* **2023**, *4*, 3100332. [CrossRef]

42. Finžgar, J.R.; Ross, P.; Hölscher, L.; Klepsch, J.; Luckow, A. Quark: A framework for quantum computing application benchmarking. In Proceedings of the 2022 IEEE International Conference on Quantum Computing and Engineering (QCE), Broomfield, CO, USA, 18–23 September 2022; pp. 226–237. [CrossRef]
43. Karalekas, P.J.; Tezak, N.A.; Peterson, E.C.; Ryan, C.A.; Da Silva, M.P.; Smith, R.S. A quantum-classical cloud platform optimized for variational hybrid algorithms. *Quantum Sci. Technol.* **2020**, *5*, 024003. [CrossRef]
44. Natu, S.; Triana, E.; Kim, J.S.; O'Riordan, L.J. Accelerate Hybrid Quantum-Classical Algorithms on Amazon Braket Using Embedded Simulators from Xanadu's PennyLane Featuring NVIDIA cuQuantum. AWS Quantum Technologies Blog. 2022. Available online: https://aws.amazon.com/blogs/quantum-computing/accelerate-your-simulations-of-hybrid-quantum-algorithms-on-amazon-braket-with-nvidia-cuquantum-and-pennylane/ (accessed on 15 July 2025).
45. Ma, L.; Leah, D. Hybrid quantum edge computing network. In *Quantum Communications and Quantum Imaging XX*; SPIE: Bellingham, WA, USA, 2022; Volume 12238, pp. 83–93. [CrossRef]
46. Wulff, E.; Garcia Amboage, J.P.; Aach, M.; Gislason, T.E.; Ingolfsson, T.K.; Ingolfsson, T.K.; Pasetto, E.; Delilbasic, A.; Riedel, M.; Sarma, R. Distributed hybrid quantum-classical performance prediction for hyperparameter optimization. *Quantum Mach. Intell.* **2024**, *6*, 59. [CrossRef]
47. Sangle, R.; Khare, T.; Seshadri, P.V.; Simmhan, Y. Comparing the orchestration of quantum applications on hybrid clouds. In Proceedings of the 2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops (CCGridW), Bangalore, India, 1–4 May 2023; pp. 313–315. [CrossRef]
48. Li, A.; Stein, S.; Krishnamoorthy, S.; Ang, J. Qasmbench: A low-level quantum benchmark suite for nisq evaluation and simulation. *ACM Trans. Quantum Comput.* **2023**, *4*, 1–26. [CrossRef]
49. Waite, W.M.; Goos, G. *Compiler Construction*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; p. 3. [CrossRef]
50. Transpiler. Available online: https://docs.quantum.ibm.com/api/qiskit/transpiler (accessed on 17 February 2025).
51. Level3.py. Available online: https://github.com/Qiskit/qiskit/blob/main/qiskit/transpiler/preset_passmanagers/level3.py (accessed on 17 February 2025).
52. Wang, Q. Optimal trace distance and fidelity estimations for pure quantum states. *IEEE Trans. Inf. Theory 2024. 70*, 8791–8805. [CrossRef]
53. Pedersen, L.H.; Møller, N.M.; Mølmer, K. Fidelity of quantum operations. *Phys. Lett. A* **2007**, *367*, 47–51. [CrossRef]
54. Hua, F.; Wang, M.; Li, G.; Peng, B.; Liu, C.; Zheng, M.; Stein, S.; Ding, Y.; Zhang, E.Z.; Humble, T.; et al. QASMTrans: A QASM Quantum Transpiler Framework for NISQ Devices. In Proceedings of the SC'23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis, New York, NY, USA, 12–17 November 2023; pp. 1468–1477. [CrossRef]
55. Mixed States; Pure States. Available online: https://pages.uoregon.edu/svanenk/solutions/Mixed_states.pdf (accessed on 17 February 2025).
56. Soeparno, H.; Anzaludin, S.P. Cloud quantum computing concept and development: A systematic literature review. *Procedia Comput. Sci.* **2021**, *179*, 944–954. [CrossRef]
57. Ferracin, S.; Hashim, A.; Ville, J.L.; Naik, R.; Carignan-Dugas, A.; Qassim, H.; Morvan, A.; Santiago, D.I.; Siddiqi, I.; Wallman, J.J. Efficiently improving the performance of noisy quantum computers. *Quantum* **2024**, *8*, 1410. [CrossRef]
58. Castelvecchi, D. IBM releases first-ever 1,000-qubit quantum chip. *Nature* **2023**, *624*, 238. [CrossRef]
59. Available online: https://aws.amazon.com/braket/ (accessed on 17 September 2024).
60. Norimoto, M.; Taku, M.; Naok, I. Quantum speedup for multiuser detection with optimized parameters in Grover adaptive search. *IEEE Access* **2024**, *12*, 83810–83821. [CrossRef]
61. Chappell, J.M.; Iqbal, A.; Lohe, M.A.; von Smekal, L.; Abbott, D. An improved formalism for quantum computation based on geometric algebra—Case study: Grover's search algorithm. *Quantum Inf. Process.* **2013**, *12*, 1719–1735. [CrossRef]
62. Bogatyrev, V.A.; Moskvin, V.S. Application of grover's algorithm in route optimization. In Proceedings of the 2023 Intelligent Technologies and Electronic Devices in Vehicle and Road Transport Complex (TIRVED), Moscow, Russia, 15–17 November 2023. [CrossRef]
63. Khanal, B.; Rivas, P.; Orduz, J.; Zhakubayev, A. Quantum machine learning: A case study of grover's algorithm. In Proceedings of the 2021 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 15–17 December 2021; pp. 79–84. [CrossRef]
64. Qiang, X.; Ma, S.; Song, H. Quantum walk computing: Theory, implementation, and application. *Intell. Comput.* **2024**, *3*, 97. [CrossRef]
65. Goldsmith, M.; Saarinen, H.; García-Pérez, G.; Malmi, J.; Rossi, M.A.; Maniscalco, S. Link prediction with continuous-time classical and quantum walks. *Entropy* **2023**, *25*, 730. [CrossRef] [PubMed]
66. Razzoli, L.; Cenedese, G.; Bondani, M.; Benenti, G. Efficient Implementation of Discrete-Time Quantum Walks on Quantum Computers. *Entropy* **2024**, *26*, 313. [CrossRef] [PubMed]

67. Samya, M.; Singh, S.S. Quantum-Social Network Analysis for Community Detection: A Comprehensive Review. *IEEE Trans. Comput. Soc. Syst.* **2024**, *11*, 6795–6806. [CrossRef]
68. Daniela, B.; Haeberlein, H.M.; Tim, C. Amazon Braket Launches New Superconducting Quantum Processor from IQM. Available online: https://aws.amazon.com/blogs/quantum-computing/amazon-braket-launches-new-superconducting-quantum-processor-from-iqm/ (accessed on 22 May 2024).