International Journal of Computational Intelligence in Engineering (IJCIE)
Vol. 01, No. 01, April 2026, pp. 17-34
ISSN: XXXX-XXXX, DOI: 10.5281/zenodo.19950102

Multi-Paradigm Architectural Taxonomy of Hybrid Quantum-Classical Learning Pipelines: From Sequential Offloading to Cloud-Native MLOps Orchestration

Aadishesh Gopal Padasalgi[1], M Krishna Prasad[2], Rajesh I S[3], Bharathi Malakareddy A[4], Geeta R B[5]

[1]Dept. of AIML, BMS Institute of Technology and Management, Bengaluru, India. Email: aadishesh05@gmail.com
[2] Dept. of AIML, BMS Institute of Technology and Management, Bengaluru, India. Email: krishnamuralimk08@gmail.com
[3] Dept. of AIML, BMS Institute of Technology and Management, Bengaluru, India. Email: rajeshaiml@bmsit.in
[4] Dept. of AIML, BMS Institute of Technology and Management, Bengaluru, India. Email: bharathi_m@bmsit.in
[5]Professor, Dept. of CSE, KLE Institute of Technology, Hubballi, Karnataka, India. Email: geetatotad@yahoo.co.in

Article Info

Article history:

Received Feb 10, 2026
Revised Apr 03,2026
Accepted Apr 11, 2026

Keywords:

Quantum-Classical Computing
Quantum Machine Learning
Variational Quantum Circuits
Cloud-Native MLOps
Computing Architectures
Cloud Quantum Computing

ABSTRACT

The combination of quantum computing and machine learning has led to the development of a new breed of hybrid quantum-classical learning pipelines that use the best features of both classical and quantum processors. Although quantum hardware is still limited by noise, small number of qubits, and short coherence times, hybrid architectures represent a realistic way towards quantum advantage in the near-term by delegating the more difficult computational parts to Quantum Processing Units (QPUs) while keeping the classical infrastructure for data preparation, optimization, and orchestration. This work surveys thoroughly design patterns and resource optimization strategies for hybrid quantum-classical learning pipelines running on clouds. Initially, a characterization of design patterns including sequential offloading, incremental replacement, parallel ensemble, and cloud-native MLOps is derived, and their appropriateness to different application domains as well as quantum resource constraints are discussed. Next, we explore resource allocation techniques ranging from static circuit partitioning to dynamic qubit-sharing, highlighting hardware-aware placement, fidelity-aware selection, and Kubernetes-based container orchestration. Then we talk about scheduling and optimization frameworks, classical schedulers converted to quantum workloads, AI-driven orchestration with deep reinforcement learning and graph neural networks, variational loop optimizations, and quantum-assisted classical scheduling. Besides the platforms, we review also open source tools and simulators. Implementation landscapes across major cloud platforms (IBM Quantum, AWS Braket) and open-source orchestrators (Qiboml Qubernetes CONQURE, HALO) are surveyed, alongside simulation environments.

Corresponding Author:

Aadishesh Gopal Padasalgi
Dept. of AIML, BMS Institute of Technology and Management, Bengaluru, India.
Email: aadishesh05@gmail.com

1. INTRODUCTION

1.1 Convergence of NISQ Computing and Cloud AI

Quantum computing has reached the phase of Noisy Intermediate-Scale Quantum (NISQ), which is mainly identified by the appearance of devices comprising tens to few hundreds of qubits that are susceptible to

decoherence and gate error, but at the same time they can run circuits that are beyond the exact classical simulation. In the meantime, artificial intelligence-with an emphasis on deep learning-has managed to take off in various fields like computer vision, natural language processing, and scientific discovery, although it requires an exponentially growing amount of computational power and energy. The fusion of these two areas has resulted in the birth of quantum machine learning (QML), which is a field that tries to use the quantum mechanical properties of superposition, entanglement, and interference to either speed up or improve the classical learning tasks.

Hybrid quantum-classical architectures have become the main trend in this area mainly because they recognize the limitations of NISQ hardware and even find ways to use them effectively instead of waiting for fault-tolerant devices. In this respect, hybrid methodologies dividing computing tasks between traditional processors (CPUs, GPUs) and quantum processing units (QPUs) can exploit the well-established classical machine learning environment such as data preprocessing, model optimization, and deployment infrastructure, etc. deciding when and how to call quantum resources for those subroutines that probably have a quantum advantage on the other hand. The community has essentially narrowed down its choices to two major branches of AI: quantum kernel methods and variational circuits (VQCs / Quantum Neural Networks) with trainability still a significant problem because of barren plateaus and hardware noise.

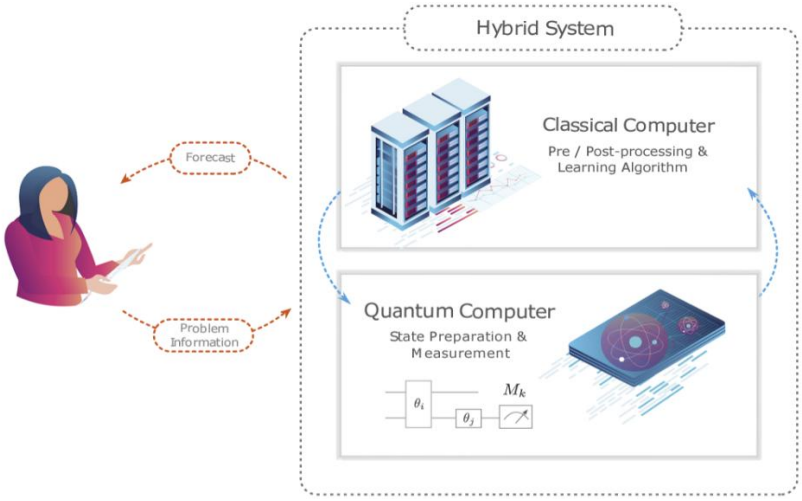


Fig. 1. Overview of a hybrid quantum-classical learning pipeline deployed in a cloud environment.

1.2 The Bottleneck: Bridging Model Design and System Efficiency

Even with a remarkable breakthrough in algorithmic aspects, there are still many systems-level issues in making hybrid QML workflows practical in a cloud setting. The main constraint is not just about algorithms. Actually, it is about how to efficiently manage different types of resources in both classical and quantum areas. Old-fashioned MLOps systems that are designed for only classical workflows cannot support the ever-changing quantum resource requirements in the pipeline-from the stage of data encoding to that of model training and finally to inference serving.

Key challenges include:

- **Resource heterogeneity:** Hybrid pipelines need to manage operations among CPU, GPU, and QPU, which are different in terms of their performance characteristics, availability times, and types of failures.
- **Scheduling complexity:** Since a quantum computer cannot be multiplexed between programs at the same time, quantum cloud platforms use fair-share schedulers. This results in qubits being idle and significant under-utilization of the hardware.
- **Classical bottleneck:** The time taken by classical post-processing (such as diagonalization based on samples in quantum diagonalization) is often far greater than that of quantum sampling, in which the classical part of computation runs for several hours and quantum part only for a few minutes.
- **Orchestration overhead:** Switching dynamically between classical and quantum computers, managing waiting times, circuit serialization, etc. add a lot of extra time.

Recent developments in cloud-native infrastructure - mainly container orchestration platforms such as Kubernetes - provide hopeful solutions to these problems by allowing dynamic distribution of quantum and classical resources, orchestration independent of paradigms, and flexible management of workflows.

1.3 Scope of the Survey and Contribution

This review covers architectural design patterns and resource optimization techniques for hybrid quantum-classical learning pipelines in cloud environments, emphasizing these deployment scenarios.

We limit our topic to:

- Tradition of architectural design patterns, from basic offloading to complex MLOps pipelines.
- Detailed study of resource allocation approaches, e.g. static circuit cutting and dynamic qubit-sharing methods.
- Thorough summary of scheduling and optimization frameworks, both classical and AI-based.
- Overview of implementation foci on major cloud platforms and open-source orchestrators. Recognizing open issues and suggesting research directions.

There are three main things that we do:

- Present the first survey that systematically connects the two worlds of QML model architecture and cloud systems engineering.
- We organize the results from more than 20 very recent papers (2024-2026) into a taxonomy that is easy to understand.
- We show which research areas need more attention and suggest a path to the integration of hybrid intelligence.

2. BACKGROUND AND FOUNDATIONAL CONCEPTS

2.1 Quantum Machine Learning Primitives

2.1.1 Variational Quantum Circuits and Parameter-Shift Rules

Variational Quantum Circuits (VQCs), also termed parameterized quantum circuits, form the computational backbone of most near-term QML algorithms. A VQC is a quantum circuit whose gates depend on tunable classical parameters $\theta \in \mathbb{R}^{d}$. The circuit prepares a quantum state $|\psi(\theta)\rangle = U(\theta)|0\rangle \otimes n$, which is then measured with respect to some observable O to yield an expectation value:

$$f(\theta) = \langle\psi(\theta)|O\ |\psi(\theta)\rangle$$

This expectation value serves as the output of the quantum model, and the parameters θ are optimized classically-typically via gradient-based methods-to minimize a loss function $L(\theta)$.

The parameter-shift rule enables exact gradient computation on quantum hardware without finite differences. For gates of the form $e^{-1}\theta P/2$ where P is a Pauli operator, the partial derivative is:

$$\partial f/\partial\theta j = (1/2)[f(\theta + (\pi/2)ej) - f(\theta - (\pi/2)ej)]$$

where ej is the unit vector along the j-th coordinate. This rule enables gradient-based optimization directly on QPUs, though at the cost of additional circuit evaluations.

2.1.2 Quantum Kernels and Data Encoding Strategies

Quantum kernel methods offer an alternative to VQCs, leveraging quantum circuits to compute inner products in high-dimensional Hilbert spaces that may be classically intractable. Given a data encoding unitary $U(x)$ that maps classical data x to a quantum state $|\varphi(x)\rangle = U(x)|0\rangle \otimes n$, the quantum kernel between two data points is:

$$k(xi, xj) = |\langle\varphi(xi)|\varphi(xj)\rangle|2$$

One way to compute this kernel is through the overlap test (also known as the SWAP test or the inversion test) and then use it with classical kernel methods like Support Vector Machines. Kernel methods, as opposed to VQCs, do not suffer from barren plateaus but still need a well-thought-out feature map design to realize quantum advantage.

The process of data encoding or how we turn classical information into quantum states plays a major role in the expressiveness and trainability of the model. A handful of popular methods are basis encoding (which is a binary representation), amplitude encoding (this is exponentially compact but producing it is quite costly), angle encoding (it involves data-driven parameterized rotations), and trainable encoding (data re-uploading). There is a growing focus in the literature on quantum-friendly data embedding that is specifically designed in accordance to hardware constraints.

2.2 Cloud-Native Computing Paradigms

2.2.1 Function-as-a-Service (FaaS) and Serverless Architectures

Serverless computing removes the need to manage infrastructure, so developers only need to focus on writing functions that run whenever called and can scale themselves. Serverless frameworks in quantum computing facilitate very detailed control over hybrid workflows: traditional classical functions take care of the data preprocessing and post-processing, while quantum functions run the quantum processing unit (QPU) execution. One great example of this approach is IBM Qiskit Serverless, which offers a managed runtime environment for hybrid quantum-classical workflows along with features like automatic queuing and result handling.

2.2.2 Container Orchestration and Kubernetes Extensions

Kubernetes is becoming the go-to platform for container orchestration in cloud settings. Custom Resource Definitions (CRDs) and operators enable the definition of quantum-specific resources through an extensible API. Qubernetes offers a single cloud-native operating environment for hybrid classic-quantum computing by considering QPUs as custom resources that can be scheduled together with CPUs and GPUs. By doing this, quantum resources can be dynamically allocated without interfering with classical MLOps workflows, keeping data and model versioning stable in hybrid environments.

2.3 The Hybrid Execution Loop: Classical Optimization over Quantum Inference

The canonical hybrid QML workflow consists of an iterative loop:

- **Classical preprocessing:** Data is normalized, dimensionally reduced, and encoded for quantum consumption.
- **Quantum execution:** The encoded data is passed to a QPU (or simulator), which executes a parameterized circuit and returns measurement results.
- **Classical post-processing:** Measurement outcomes are aggregated, expectation values computed, and loss gradients evaluated.
- **Parameter update:** A classical optimizer updates the circuit parameters θ.
- Repeat until convergence.

This loop introduces unique orchestration challenges: the quantum step is subject to queuing delays and device availability windows, while the classical optimization step may run continuously on CPU/GPU infrastructure. Efficient execution requires decoupling these components and managing asynchronous communication.

2.4 Review Methodology and Paper Selection Criteria

This research review integrates the literature from 2024 to 2026 published mainly in Nature portfolio journals, APS Physical Review, IEEE Transactions, ACM conferences (SC ICPP CCGrid), and preprint servers (arXiv).

We performed a systematic search centered on the nexus of the following terms:

- Hybrid Quantum-Classical
- Machine Learning
- Cloud Computing
- Resource Allocation
- Scheduling

Articles were primarily picked because of:

- New architectural contributions
- Experiments conducted on actual quantum hardware or realistic simulators
- Source code publicly available

- Suitability for cloud deployment

Studies dealing exclusively with algorithms and lacking systems perspective, as well as opinion articles without verifiable claims were left out.

Table 1 is a well-organized summary of the 20 main documents that have been a focus of this review.

Table 1. Summary of reviewed literature

Ref	Year	Design Pattern	Resource Allocation	Scheduling Approach	Implementation	Key Metric
[1]	2025	Incremental Substitution	Layer-wise classical-to-quantum replacement	Not addressed	Python simulators	Classification accuracy
[2]	2026	Modular block design	Frozen classical feature extraction + trainable quantum	Not addressed	Noise-aware simulation	Generalization bound
[3]	2024	Backend resource management	Adapts CPU scheduling to QPU	QSRA scheduling algorithm	Simulation	Turnaround time
[4]	2025	Distributed scheduling	Multi-QPU orchestration	RL-based adaptive	Simulation	Runtime, fidelity
[5]	2024	Serverless function orchestration	RL-driven device allocation	GCN + RL	95 circuits on 10 IBM devices	Completion time
[6]	2025	Hybrid optimization	Offloads combinatorial optimization to QPU	QAOA + VQE	Simulator	Energy, makespan
[7]	2026	Quantum RL + metaheuristic	VQC policy encoding	Golden Jackal Optimization	EdgeBench IoT	Latency, energy
[8]	2025	FaaS workflow patterns	Hybrid cloud partitioning	Adaptive polling	Hybrid cloud	Throughput
[9]	2026	MLOps pipeline	Dynamic container allocation	Kubernetes orchestration	Kubernetes	Flexibility, reproducibility
[10]	2025	Orchestration library	Manages quantum-classical communication	Orchestration layer	Qibo + TF/PyTorch	Backend flexibility
[11]	2024	Task placement	DRL agent for QPU selection	DRL	Simulated quantum cloud	Completion time
[12]	2025	QAI resource management	Simulates allocation policies	Quantum RL	Python simulation	Policy evaluation
[13]	2025	Fidelity-aware orchestration	DRL-based task allocation	DRL	QSimPy	Fidelity, speed
[14]	2020	Platform-level design	Cloud optimization	Parametric compilation	Quantum Cloud Services	Runtime
[15]	2025	Serverless system model	QPU and classical function execution	Trace-driven simulation	IBM Qiskit serverless	Queue metrics
[16]	2024	Cloud-native execution	Kubernetes resource management	Kubernetes scheduling	Kubernetes	Scalability
[17]	2024	Orchestrator pattern	Hybrid scheduler	QoS balancing	Hybrid cluster	Completion time, utilization
[18]	2025	Co-design framework	Classical HPC + quantum integration	Not applicable	Framework-level	Dependability
[19]	2025	Hybrid architecture	Quantum for optimization subroutines	Not detailed	Hybrid cloud	Clustering efficiency
[20]	2025	Survey of hybrid workflows	Not applicable	Not applicable	Various frameworks	N/A

3. A TAXONOMY OF ARCHITECTURAL DESIGN PATTERNS

3.1 Overview of the Pattern Landscape

Hybrid quantum-classical architectures come with various structural patterns that essentially illustrate different compromises or trade-offs among the consumption of quantum resources, classical computation, and communication overhead. We distinguish four main design patterns, in fact, the four basic design patterns based on the degree of sophistication and integration depth are:

- **Sequential Offloading and Feature Extraction:** Quantum processors serve as drop-in accelerators for specific subroutines, with clear separation between classical pre/post-processing and quantum execution.
- **Incremental Substitution (Creeping Quantum):** Classical neural network components are progressively replaced by quantum equivalents, enabling fine-grained analysis of quantum utility.
- **Parallel Ensemble and Asynchronous Pipelines:** Multiple classical and quantum models execute concurrently, with voting or aggregation mechanisms combining their outputs.
- **Cloud-Native MLOps Workflows:** Full lifecycle management of hybrid models using container orchestration, serverless functions, and automated pipeline composition.

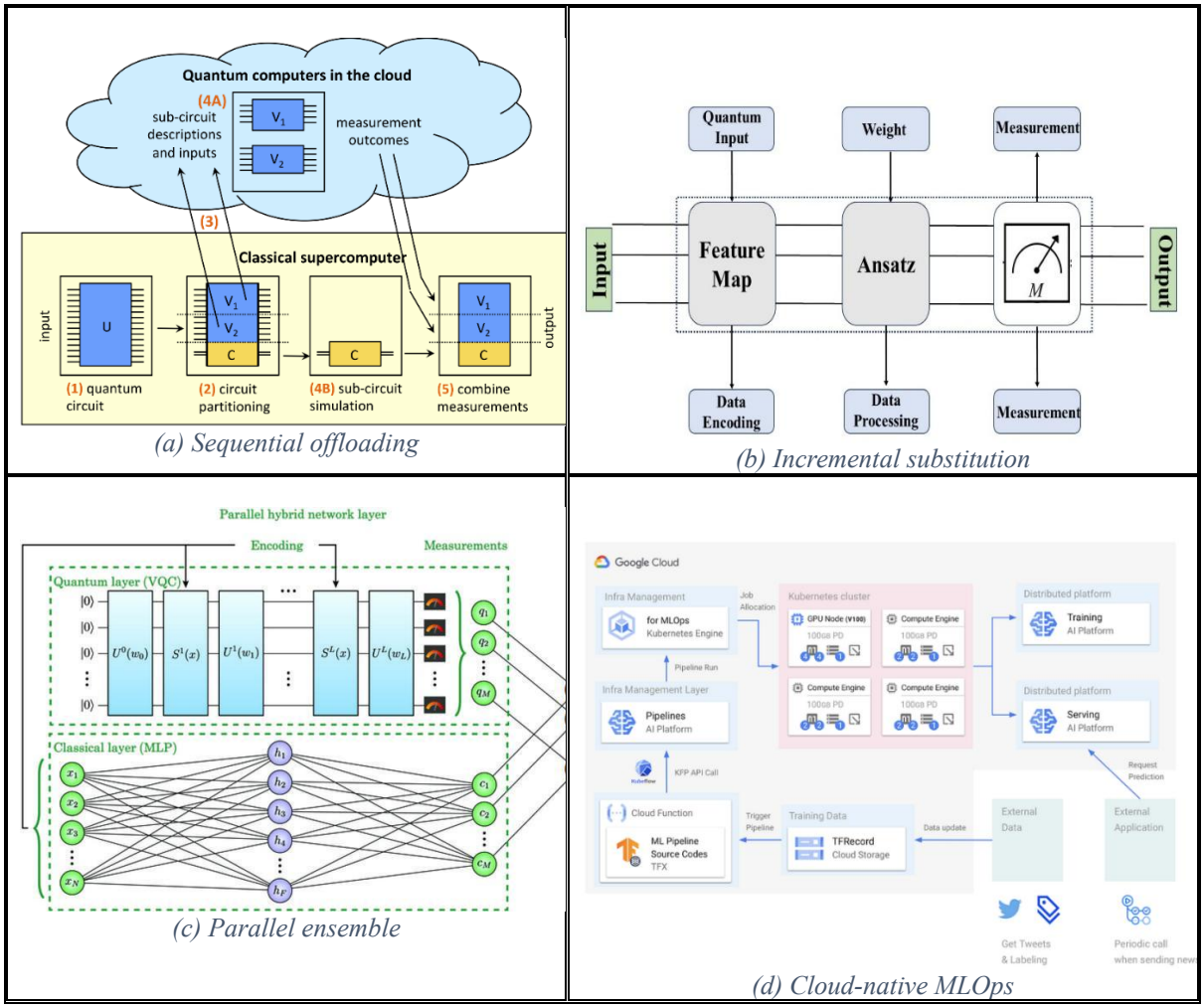


Fig. 2. Taxonomy of architectural design patterns.

(a)- https://www.epiqc.cs.uchicago.edu/hybrid-quantum-classical-computing,

(b)- https://www.mdpi.com/2227-7390/12/23/3684,

(c)- https://thequantuminsider.com/2023/10/30/parallel-hybrid-network-achieves-better-performance-through-quantum-classical-collaboration/,

(d)- https://medium.com/google-developer-experts/mlops-big-picture-in-gcp-a637566d6ae8

3.2 Pattern I: Sequential Offloading and Feature Extraction

This approach, which is a fairly simple one and is actually the predominant one, sees the quantum-processing-unit (QPU) as a co-processor that gets invoked during certain points of the classical pipeline. Firstly, a classical architecture (e.g. ResNet VGG) is used to obtain high-level features from the raw data. The obtained features are then dimensionally reduced and converted into a quantum circuit. Then, the quantum-processing-unit (QPU) 'runs' the encoded features (e.g. through a variational quantum circuit (VQC) or kernel evaluation). The results of measurement subsequently determine the final class or regression output.

Representative Work: Post Variational Classical Quantum Transfer Learning (PVCQTL) is a method that enhances VQC capabilities in transfer learning, by implementing post-variational tactics that not only cut down on training time but also help to improve optimization challenges. PVCQTL consists of three configurations: a re-engineered observable construction, a hybrid, and a variational-post-variational combination. Testing these on pre-trained models (VGG19 ResNet50 ResNet18, MobileNet) for the deepfake detection task resulted in 85% accuracy along with the decreased computational expenses.

Pattern Strengths:

- Minimal modification to existing classical ML pipelines.
- Clear separation of concerns simplifies debugging and performance profiling.
- Quantum resource consumption is bounded and predictable.

Pattern Limitations:

- Communication overhead between classical and quantum components.
- Limited opportunity for end-to-end quantum advantage; gains are confined to the quantum-accelerated submodule.

3.3 Pattern II: Incremental Substitution (Creeping Quantum)

Incremental substitution gradually replaces the components of a classical neural network such as single layers, activation functions or entire blocks with quantum counterparts while keeping the overall design unchanged. This approach makes it possible to do precise ablation studies that measure the additional effect of quantum components.

Representative Work: The Let the Quantum Creep In framework recommends a detailed layer-wise classical-to-quantum replacement, showing performance variation on MNIST, FashionMNIST, and CIFAR-10. The framework allows the gradual change of granularity at multiple points, from individual neurons to whole convolutional blocks.

Pattern Strengths:

- Granular control over quantum resource consumption.
- Empirical isolation of quantum utility, facilitating principled design decisions.
- Compatibility with existing pre-trained classical models.

Pattern Limitations:

- Exponential search space for optimal substitution configuration.
- Interface mismatches between classical activation functions and quantum measurement statistics.

3.4 Pattern III: Parallel Ensemble and Asynchronous Pipelines

Parallel ensemble architectures are capable of running multiple classical and quantum models side by side, and then combining their results through voting, averaging, or using a meta-classifier that could be learned. This pattern takes into consideration the latency and availability limitations of cloud QPUs: while classical models offer a starting point for predictions, quantum models run in a non-blocking manner and add their inputs when their outcomes are ready.

Representative Work: The Hybrid quantum-classical reinforcement learning (HyRL) framework uses a quantum neural network as the Actor to generate policies and take advantage of quantum exploration, whereas a classical neural network is used as the Critic to assess the actions. Such segmentation achieves a synergistic balance and boosts the training efficiency, which is shown in the example of cloud task scheduling.

Pattern Strengths:

- Robustness to QPU unavailability or queuing delays.
- Graceful degradation: classical fallback ensures continuous service.
- Opportunity for speculative execution and result fusion.

Pattern Limitations:

- Increased computational cost from redundant execution.
- Complex synchronization logic for result aggregation.

3.5 Pattern IV: Cloud-Native MLOps Workflows

Cloud-native MLOps patterns involve the extension of traditional MLOps principles-continuous integration, continuous deployment, model versioning, and monitoring-to the hybrid quantum-classical pipelines. Containerized microservices encapsulate quantum circuit execution, classical optimization, and data processing and are orchestrated via Kubernetes or serverless frameworks.

Representative Work: The Cloud-Native MLOps Framework makes use of container orchestration to back quantum-classical ML workflows through the dynamic provision of quantum and classical resources, orchestration without being tied to any specific paradigm, as well as flexible management of workflows. Since it is based on Kubernetes, it is the framework that allows enabling the flexible provision of quantum resources at any stage of the pipeline even without the interruption of classical MLOps workflows.

Pattern Strengths:

- Reproducible, versioned pipelines across hybrid environments.
- Automatic scaling of classical components based on load.
- Unified observability and logging across classical and quantum resources.

Pattern Limitations:

- Significant infrastructure complexity and operational overhead.
- Immature ecosystem of quantum-specific Kubernetes operators.

3.6 Comparative Analysis of Pattern Suitability

Table 2 summarizes the suitability of each pattern across key dimensions.

Table 2. Comparative analysis of architectural design patterns

Dimension	Sequential Offloading	Incremental Substitution	Parallel Ensemble	Cloud-Native MLOps
Quantum resource efficiency	Moderate	Low (Multi evals)	High (selective)	Configurable
Classical integration complexity	Low	Moderate	High	Very high
Fault tolerance	Low	Low	High	Moderate
Latency sensitivity	High	Moderate	Low	Configurable
Reproducibility	Moderate	Moderate	Moderate	High
Suitable for NISQ era	Excellent	Good	Excellent	Good
Production readiness	Moderate	Low	Moderate	High (emerging)

4. RESOURCE ALLOCATION: MANAGING CPU AND QPU ASSETS

4.1 Static Circuit Partitioning and Circuit Cutting

Static resource allocation means to divide quantum circuits at the time of compilation and then distribute the subcircuits to several QPUs or to allocate certain parts of the qubit subsets within a single device. Circuit cutting is a technique of decomposing a large circuit into smaller pieces that can be independently executed. It allows for running the QPU with limited qubit counts though it requires classical post-processing to get the full circuit result.

Representative Work: Deadline-Aware Scheduling of Distributed Quantum Circuits presents a distributed quantum circuits (DQC) scheduling optimization problem that is solved by simulated annealing. The framework models interdependencies among subcircuits after the partitioning and collectively allocates sampling shots across quantum processing units (QPUs) for efficient wire cutting. Experimental results indicate a 12.8% uplift in the number of requests served within deadlines and 8.16% additional requests served in comparison with dependency-agnostic baselines.

Static Allocation Considerations:

- **Communication overhead:** Subcircuits must exchange classical information proportional to the number of cuts.
- **Fidelity degradation:** Each cut introduces reconstruction error that compounds with circuit depth.
- **Topology constraints:** Qubit allocation must respect device connectivity graphs, minimizing SWAP gate overhead.

4.2 Dynamic Allocation Heuristics and Qubit Reuse

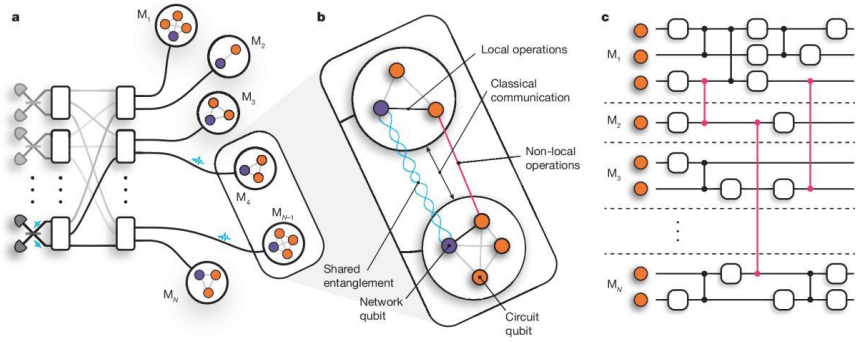
Dynamic allocation changes resource assignments on the fly by taking into account device availability, queue lengths, and job priorities. It's especially important in multi-tenant quantum clouds where tens of thousands of users share only a few quantum processors.

Representative Work: The only quantum operating system design capable of supporting fine-grained resource-sharing, according to HALO, is the first one. HALO comes up with two methods: a hardware-aware qubit-sharing way that aims at arranging shared assistant qubits in order to reduce the routing overhead and cross-talk noise, and a shot-adaptive scheduler that assigns the time when each job runs based on its sampling needs. The study conducted on IBM Torino demonstrates that HALO can enhance hardware usage by 2.44 times, increase throughput by 4.44 times, and restrict fidelity loss to 33% compared to HyperQ.

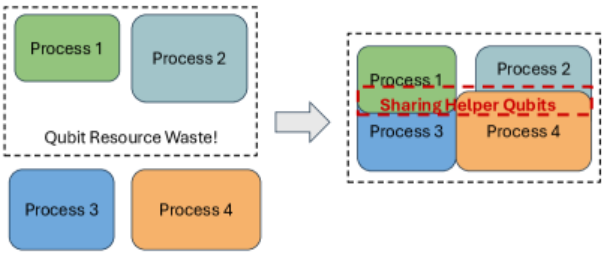
Representative Work: Virtual QPU or VQPU for short supports quantum cloud computing to work at its full capacity by promising the circuit fidelity. VQPU handles the circuit compilation, it is also self-driven to generate virtual QPUs which it then gives to the related tasks. Thus, multiplexing of physical qubits is done in an efficient way.

Dynamic Allocation Challenges:

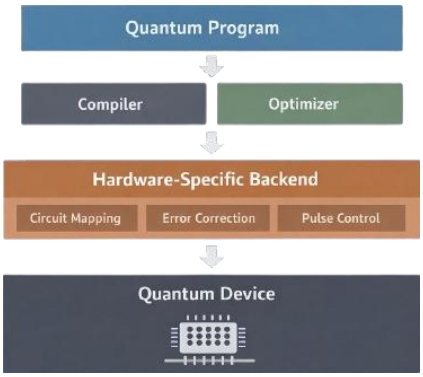
- **State preservation:** Qubit states must be stored and restored across context switches, introducing decoherence risk.
- **Crosstalk mitigation:** Simultaneous execution on adjacent qubits can induce correlated errors.
- **Fairness vs. efficiency:** Balancing equitable access with maximal hardware utilization.



(a) Static partitioning with circuit cut



(b) Dynamic qubit-sharing with HALO



(c) Infrastructure-aware selection across heterogeneous QPUs.

Fig. 3. Comparison of resource allocation strategies.

(a)- https://www.nature.com/articles/s41586-024-08404-x

(b)- https://arxiv.org/html/2602.07191

(c)- https://everydaysworld9980.medium.com/design-and-implementation-of-a-quantum-computing-system-fd49a99c1240

4.3 Infrastructure-Aware Selection in Heterogeneous Clouds

Quantum cloud platforms (IBM AWS IonQ) provide different types of QPUs that differ in the number of qubits they have, their connectivity, gate fidelities, and queue lengths. Infrastructure-aware selection makes a decision on the best QPU for each job by analysing the circuit characteristics and considering user-defined constraints (fidelity requirements, cost budgets, deadlines).

Representative Work: Qonductor offers a hardware-agnostic API for hybrid applications along with a hybrid scheduler that balances QoS and resource efficiency. The orchestrator reaches 54% less job completion time and 66% higher utilization compared to baseline schedulers.

Representative Work: CONQURE gives you an entirely open-source cloud queue framework alongside a modular scheduling framework that allows OpenMP quantum kernels to be offloaded to QPUs as quantum circuits. The CONQURE API has an overhead averaging of only 12.7ms, and the OpenMP extension achieved a 3.1 reduction in VQE runtime.

Infrastructure-Aware Metrics:

- **Circuit depth tolerance:** Maximum depth before decoherence overwhelms signal.
- **Connectivity match:** How well the circuit's two-qubit gate pattern aligns with device topology.
- **Calibration recency:** Freshly calibrated devices exhibit higher and more stable fidelities.
- **Queue wait time:** Predicted time until execution based on current queue length and job priorities.

5. SCHEDULING AND OPTIMIZATION FRAMEWORKS

5.1 Classical Schedulers Adapted for Quantum Workloads

Quantum scheduling problem structurally resemble classical job scheduling yet also bring in additional constraints that qubit decoherence forces scheduling with strict time windows, gate errors are more when circuit depth increases, and measurement results are non-deterministic. Classical scheduling heuristics like First-Come-First-Served (FCFS), Shortest-Job-First (SJF), Priority Queuing, and Fair-Share were modified to meet quantum requirements and then used.

Representative Work: QSRA borrows CPU scheduling techniques and applies them to quantum processing units (QPUs). This helps to increase qubit utilization and decrease turnaround time. The method consists of a qubit allocation subroutine which takes into account quality and connectivity, and a merge of several quantum programs to optimize usage.

Limitations of Classical Adaptation:

- Classical schedulers assume deterministic job durations, whereas quantum circuit execution time depends on shot count and optimization convergence.
- Fair-share schedulers, employed by current quantum cloud platforms, leave many qubits idle and significantly under-utilize hardware.
- Lack of preemption support: quantum circuits cannot be paused and resumed without state loss.

5.2 AI-Driven Orchestration Techniques

5.2.1 Deep Reinforcement Learning for Task Placement

Deep Reinforcement Learning (DRL) is being recognized as a method with tremendous potential for adaptive scheduling in environments that are complicated and constantly changing. DRL agents can figure out the best task placement policies by experimenting with simulated or real quantum cloud environments and aiming at metrics like job turnaround time, efficient use of resources, and fidelity.

Representative Work: DRLQ introduces a method that is based on Deep Reinforcement Learning (DRL) for the placement of tasks in quantum cloud environments. The agent is trained to assign quantum tasks to the available QPUs, such that the time taken for the completion of the tasks and the efficiency of scheduling are maximized.

Representative Work: Using a deep reinforcement learning agent QFOR a quantum cloud quantum fidelity aware orchestrator is presented, which is balancing the speed and accuracy in heterogeneous quantum cloud environments. Based on the QSimPy simulation environment, QFOR prioritizes not only the fidelity but also the speed of the execution.

DRL Formulation: The scheduling problem is modeled as a Markov Decision Process (MDP):

- **State:** Current queue lengths, device availability, circuit characteristics.

- **Action:** Assignment of pending jobs to specific QPUs or queue positions.
- **Reward:** Weighted combination of negative completion time, positive fidelity, and resource utilization.

5.2.2 Graph Neural Networks for Circuit Deployment

Graph Neural Networks (GNNs) effectively model the inherent topological structure of quantum circuits (nodes represent qubits and edges indicate two-qubit gates) as well as of the device connectivity graphs. Schedulers based on GNNs acquire embeddings which are utilized for determining the placement and routing of circuits.

Representative Work: Moirai introduces a quantum serverless function orchestration system by means of reinforcement learning with GCN embedding to represent circuits. Compared to the baselines, Moirai not only lessens the maximum completion time but also enhances the usage of devices by more than 30%. The testing on 95 quantum circuits (3-7 qubits) carried out on 10 IBM quantum devices reveals the capability of the method to work efficiently with different chip topologies.

GNN Embedding Approach: A Graph Convolutional Network (GCN) processes the circuit's interaction graph to produce a compact vector representation capturing:

- Gate type distribution and connectivity patterns.
- Critical path length and parallelism opportunities.
- Qubit interaction density and locality.

This embedding informs device selection and qubit placement decisions.

5.2.3 Multi-Agent Systems and Edge Coordination

Coordination of edge-cloud introduces the idea of complexity. This is due to the need of distributed scheduling at the QPUs that are geographically dispersed and classical nodes. Multi-agent reinforcement learning (MARL) can be an aid for decentralized decision-making when each agent only has local observability.

Representative Work: QRGEC combines quantum-centric policy search (through VQCs) with adaptive metaheuristic parameter tuning (Golden Jackal Optimization) for an efficient edge-cloud collaboration. This system manages to improve latency by 36.8%, energy by 24.7%, fault tolerance by 48.2%, and 94% resource usage on edge-cloud test sets.

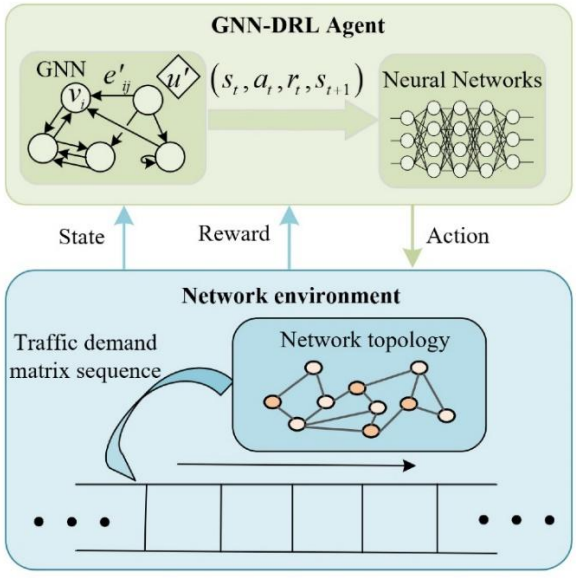


Fig. 4. Overview of AI-driven scheduling framework (https://www.mdpi.com/2079-9292/11/18/2952)

DRL agent processes state and outputs placement actions. GNN encoder generates circuit embeddings from gate topology. Reward function balances completion time, fidelity, and utilization.

5.3 Variational Loop Optimizations

The iterative nature of variational algorithms creates unique optimization opportunities within the hybrid execution loop.

5.3.1 Parametric Compilation and Active Qubit Reset

Parametric compilation stores the compiled circuit templates, and only changes the parameter-dependent gate angles between iterations; it does not recompile operations, which is a quite expensive process. Active qubit reset prepares qubits to the ground state much more quickly than waiting for natural relaxation (T1 decay), that is, reduce inter-shot latency.

Representative Work: The Quantum-Classical Cloud Platform set up for variational hybrid algorithms introduces not just performance measurement framework but also platform changes such as parametric compilation and active qubit reset. These improvements minimize the delay and enhance the production capacity of variational algorithms executing on Quantum Cloud Services.

5.3.2 Mitigating Queue Delays via Adaptive Polling

Adaptive polling methods change how often they check statuses depending on the anticipated wait times in the queue, so they download less information over the network but still remain responsive. Predictive models that have learned from past queue data predict when the execution will start, thereby allowing just-in-time classical preparation.

5.4 Quantum-Assisted Classical Scheduling

A meta-pattern emerges when quantum algorithms themselves optimize classical scheduling decisions. This recursive relationshipquantum computers scheduling their own classical support infrastructureillustrates the deep hybrid systems intertwining.

Representative Work: The Quantum-Assisted Framework for Energy-Efficient Cloud Task Scheduling relies on QAOA to find near-optimal task-to-resource allocations and VQE to improve the energy model. The framework is able to save 23.6% energy, cut the makespan by 17.4%, and improve resource utilization by 17.7% when compared to genetic algorithms and particle swarm optimization.

Algorithm Integration: Classical Scheduler → [problem encoding] → QAOA → [candidate solutions] → VQE Refinement → [final schedule] → Execution.

This approach offloads combinatorial optimization to QPUs while retaining classical heuristics for fine-tuning.

6. IMPLEMENTATION LANDSCAPES AND PLATFORM INTEGRATION

6.1 Major Cloud Quantum Platforms

6.1.1 IBM Quantum Services

IBM Quantum provides the most comprehensive cloud quantum ecosystem, encompassing:

- **Qiskit Runtime:** A managed execution environment that co-locates classical computation with quantum hardware, reducing communication latency. Runtime primitives (Sampler, Estimator) abstract away circuit execution and error mitigation.
- **Qiskit Serverless:** A FaaS platform for hybrid quantum-classical workflows, supporting automatic queuing, result storage, and parallel execution across multiple QPUs.
- **IBM Quantum Platform:** Access to 20+ QPUs ranging from 5 to 1,121 qubits, with varying topologies (heavy-hex, Falcon) and gate fidelities.

6.1.2 AWS Braket Hybrid Jobs

AWS Braket provides a single programming interface that spans several QPU vendors (IonQ, Rigetti, OQC, QuEra) and managed simulators (SV1, TN1, DM1). Hybrid Jobs allow classical and quantum computations to be combined in one execution environment, with classical compute instances being provisioned automatically and data storage integrated with Amazon S3. Braket's quantum-synchronized operating layer unifies the interface for quantum experiments and hybrid workflows.

6.1.3 Emerging Platforms

- **Qoro Solo:** A self-serve cloud platform providing streamlined access to classical simulation of hybrid quantum-classical workloads, scaling across CPUs, GPUs, and QPUs without extensive custom integration.
- **Qilimanjaro SpeQtrum QaaS:** A tri-modal architecture supporting quantum workflows in simulation, optimization, and AI model training.
- **IonQ Quantum OS:** A new quantum operating system and hybrid services suite designed for enterprise customers.

6.2 Open-Source Orchestrators and Simulation Toolkits

6.2.1 Qiboml

Qiboml is a free to use Python library that makes it easy to control hybrid machine learning workflows where quantum and classical parts come together. Qiboml harnesses Qibo's quantum computing features & extends them by working alongside TensorFlow and PyTorch so that you can design a model that will operate in multi-threaded CPUs, GPUs, and multi-GPU systems for simulation with statevector or tensor network methods; quantum processing units, both on-premise and through cloud providers; and noise-aware simulations and real-time error mitigation and calibration.

6.2.2 Qubernetes

Qubernetes extends Kubernetes with Custom Resource Definitions (CRDs) for quantum resources, enabling unified scheduling of classical and quantum workloads. The platform treats QPUs as schedulable resources alongside CPU and GPU, allowing existing Kubernetes tooling (Helm charts, Operators) to manage hybrid deployments.

6.2.3 CONQURE

CONQURE (Co-execution Environment for Quantum and Classical Resources) provides a fully open-source cloud queue framework with OpenMP extensions for quantum kernel offloading. The API enables offloading OpenMP quantum kernels to QPUs as quantum circuits, relaying results back to calling contexts in classical computing, and scheduling quantum resources via the CONQURE API. The API overhead averages only 12.7ms, and OpenMP extension enables 3.1× reduction in VQE runtime on ion-trap devices.

6.2.4 HALO

HALO (Fine-Grained Resource Sharing Quantum Operating System) introduces hardware-aware qubit-sharing, which places shared helper qubits to minimize routing overhead and avoid cross-talk noise, and shot-adaptive scheduling, which allocates execution windows based on each job's sampling requirements. On IBM Torino, HALO improves utilization by up to 2.44× and throughput by 4.44×, with fidelity loss within 33% compared to HyperQ.

6.2.5 Simulation Environments

- **QAISim:** A Python toolkit for modeling and simulation of AI-driven resource management policies in quantum cloud environments, using quantum RL with parameterized circuits.
- **Digital Twin Framework:** A SimPy-based discrete-event simulation replicating quantum cloud environments including device modeling, job lifecycle management, and noise-aware fidelity estimation. Supports distributed scheduling and concurrent execution on networked QPUs connected via real-time classical channels.
- **Modeling Framework for Serverless Hybrid Platforms:** An open-source numerical simulator that models hybrid classical-quantum serverless systems, driven by production traces and evaluating heterogeneous job sizes and priorities.

6.3 Representative Case Studies

6.3.1 Case Study 1: Energy-Efficient Cloud Task Scheduling (VQE + QAOA)

Problem: Allocate cloud tasks to heterogeneous resources minimizing energy consumption while meeting deadlines.

Hybrid Solution:

- QAOA explores the combinatorial space of task-to-resource assignments, generating near-optimal candidate mappings.
- VQE refines the energy model by variationally optimizing a Hamiltonian encoding power consumption.
- Classical post-processing selects the final schedule based on refined energy estimates.

Results: We got 23.6% less energy, 17.4% shorter time to finish, 17.7% more resource utilization by doing these things than by using classical metaheuristics (GA, PSO, ACO). The pace of convergence got better by 36.6%.

Key Insight: By combining quantum and classical methods for optimization, it is possible to get better results than only classical methods in scheduling problems that are NP-hard, especially when the quantum part deals with the combinatorial explosion and the classical part deals with continuous refinement.

6.3.2 Case Study 2: Resilient Edge-Cloud Coordination via VQC-based RL

Problem: Coordinate computation across edge devices and cloud resources under dynamic network conditions and workload fluctuations.

Hybrid Solution:

- Variational Quantum Circuit (VQC) encodes the system state and generates policies for task offloading decisions.
- Golden Jackal Optimization adaptively tunes hyperparameters of the RL agent.
- Classical components execute the selected actions and provide reward signals.

Results: 36.8% latency reduction, 24.7% energy efficiency improvement, 48.2% resilience enhancement, 94% resource utilization.

Key Insight: VQCs can effectively explore high-dimensional policy spaces in RL, while metaheuristic tuning accelerates convergence and improves robustness to environment shifts.

6.3.3 Case Study 3: GPU-Accelerated Classical Post-Processing for SQD

Problem: In Sample-based Quantum Diagonalization (SQD) for quantum chemistry, classical diagonalization dominates runtime-taking hours while quantum sampling requires minutes.

Hybrid Solution:

- Quantum processor samples electronic configurations from a parameterized circuit.
- Classical GPU-accelerated system performs diagonalization of the reduced Hamiltonian.
- OpenMP offloading distributes the workload across multiple GPUs.

Results: GPU-offloaded approaches reduce classical post-processing from hours to minutes, narrowing the gap between classical processing and quantum execution.

Key Insight: The classical bottleneck in hybrid algorithms demands careful engineering of classical components-not just quantum circuit optimization. GPUs are becoming core infrastructure for hybrid quantum computing.

7. OPEN CHALLENGES AND FUTURE RESEARCH DIRECTIONS

7.1 System-Level Gaps

7.1.1 The Cold Start Problem in Quantum Serverless Computing

When serverless functions are initiated on-demand, they face cold-start latency issue. The problem becomes even more serious in quantum contexts:

- QPU calibration overhead: Devices require periodic calibration that can take minutes to hours.
- Circuit compilation time: Compiling high-level quantum programs to device-native gates is computationally intensive.
- Classical environment setup: Provisioning classical compute instances and loading ML models adds latency.

Future Direction: Warm-start tactics entail pre-provisioning quantum circuit templates and keeping classical execution environments alive for the most frequently used hybrid functions.

7.1.2 State Management and Circuit Serialization Overhead

Hybrid pipelines need to be able to convert circuit definitions and measurement data into a serialized format to enable communication between classical and quantum components. Existing serialization formats like OpenQASM, QIR result in very high overhead when dealing with large circuits. Furthermore, the problem of quantum state preservation in the event of a classical context switch has not been resolved yet which means that true preemptive multitasking on QPUs is prohibited.

Future Direction: Efficient binary serialization formats and hardware-supported state snapshot mechanisms enabling checkpointing and migration of quantum computations.

7.2 Optimization Trade-offs

7.2.1 Fidelity vs. Throughput

Resource-sharing schemes such as HALO can enhance throughput however they lead to a drop in fidelity due to crosstalk and decoherence. A key issue is rigorously measuring this trade-off and providing users with a means to indicate their permissible fidelity reduction, which is vital for real-world use.

Future Direction: Fidelity-aware SLAs (Service Level Agreements) that allow users to trade execution speed for result quality, with pricing models reflecting the chosen fidelity tier.

7.2.2 Fairness and Multi-Tenancy in Shared QPU Access

When quantum clouds will have thousands of users then fairness mechanisms will have not only a balanced equal access but also efficient hardware usage. For example, fair-share schedulers keep qubits unoccupied, and dynamic multiplexing can cause competition and make the qubits less faithful.

Future Direction: Market-based scheduling mechanisms (e.g., auction-based allocation) that allow users to bid for priority access, combined with fairness guarantees to prevent starvation.

7.3 Preparing for Fault-Tolerant Hybrid Integration

7.3.1 Transitioning Hybrid Pipelines to Logical Qubits

Most hybrid QML studies are based on physical qubits with rather high error rates. The move, at some point, to fault-tolerant logical qubits will make the cost-benefit analysis different from the ground up: on the one hand, the constraints on the circuit depth will be less stringent; however, on the other hand, the number of the first logical qubits available will be very limited.

Future Direction: Co-design of QML algorithms and error correction codes to optimize the trade-off between logical qubit overhead and algorithmic accuracy.

7.3.2 Dependable System Engineering for QHPC Integration

Integrating quantum accelerators into high-performance computing (HPC) environments requires dependable system engineering addressing:

- Coordinated scheduling across classical HPC job queues and quantum cloud queues.
- Classical-quantum interaction management, including asynchronous result handling.
- Support for diverse backend resources (simulators, emulators, physical QPUs) in heterogeneous systems.

Frameworks like Q-IRIS demonstrate proof-of-concept hybrid execution integrating the IRIS asynchronous task-based runtime with the XACC quantum programming framework.

Future Direction: Standardized interfaces between classical HPC schedulers (Slurm, PBS) and quantum resource managers, enabling seamless hybrid job submission and monitoring.

7.3.3 Toward Seamless Hybrid Intelligence

The ultimate vision is seamless hybrid intelligence: computational systems that dynamically allocate tasks across classical and quantum resources without explicit programmer intervention. Achieving this requires:

- **Intelligent compilers:** Automatically partition programs into classical and quantum segments based on cost models.
- **Adaptive runtimes:** Monitor execution and dynamically reallocate resources in response to device availability and performance.
- **Unified programming models:** Abstract away the distinction between classical and quantum execution, exposing a single coherent interface to developers.

While substantial challenges remain, the rapid progress in cloud-native orchestration, AI-driven scheduling, and hardware-aware compilation suggests that seamless hybrid intelligence may be achievable within the next decade.

8. CONCLUSION

Through this survey, a thorough exploration of hybrid quantum-classical learning pipelines has been achieved from a systems perspective, thereby connecting the dots between quantum machine learning algorithms and the cloud deployment infrastructure. We have developed a classification of architectural design patterns-sequential offloading, incremental substitution, parallel ensemble, and cloud-native MLOps-and examined their appropriateness given different quantum resource limitations and application needs. Furthermore, we have highlighted the conflict between fidelity preservation and throughput maximization by addressing resource allocation strategies through static circuit partitioning, dynamic qubit-sharing, and infrastructure-aware selection. We also identified a growing inclination towards AI-driven orchestration in scheduling frameworks as deep reinforcement learning and graph neural networks facilitate adaptive, fidelity-aware task placement in heterogeneous quantum clouds. The implementation landscape is evolving quite fast with major cloud providers (IBM, AWS) releasing managed hybrid services and open-source orchestrators (Qiboml Qubernetes CONQURE, HALO) coming up with flexible alternatives.

Nevertheless, some challenges are still large and open after even with all this advancement. Quantum serverless computing's cold start issue, managing the state changes across classical and quantum boundaries, and the trade-offs between fidelity and throughput in multi-tenant environments are only a few of the areas that need research. Co-designing algorithms, error correction, and system architecture will be ways to prepare hybrid pipelines when fault-tolerant logical qubits finally become our starting point. Also, moving closer to an uninterrupted hybrid intelligence whereby computational tasks are not only dynamically but also automatically divided between classical and quantum resources is going to be a new frontier.

Additionally, the integration of quantum computing, cloud infrastructure, and artificial intelligence will be a total upheaval of computation rather than a simple step forward. As quantum devices get better and the tools for orchestrating cloud-native keep changing, we can expect that, more and more, the hybrid quantum-classical learning pipelines are going to be moved out of research labs and into production environments. This paper is a well organized manual to the researchers and professionals who are working within this transformational crossroad, giving at the same time a geographical chart of the present area and an instrument to guide for new discoveries.

CONFLICT OF INTEREST STATEMENT

The authors declare no conflict of interest.

DATA AVAILABILITY

Data can be provided on genuine request.

REFERENCES

[1] P. Wang, C. R. Myers, L. C. L. Hollenberg, and U. Parampalli, "Let the quantum creep in: Designing quantum neural network models by gradually swapping out classical components," Quantum Mach. Intell., vol. 7, no. 1, p. 62, 2025.

[2] J. Qi, X. Wang, Y. Li, L. Zhang, S. Kumar, and M. Chen, "Quantum LEGO learning: A modular design principle for hybrid artificial intelligence," arXiv preprint arXiv:2601.21780, 2026.

[3] B. Lu, Z. Chen, and Y. Wu, "QSRA: A QPU scheduling and resource allocation approach for cloud-based quantum computing," arXiv preprint arXiv:2411.05283, 2024.

[4] W. Luo, S. K. Singh, A. Paler, J. M. Martinis, R. Biswas, and K. Bertels, "Adaptive job scheduling in quantum clouds using reinforcement learning," in Proc. 54th Int. Conf. Parallel Process. (ICPP), 2025, pp. 1–10.

[5] T. Li and Z. Zhao, "Moirai: Optimizing quantum serverless function orchestration via device allocation and circuit deployment," in Proc. IEEE Int. Conf. Web Services (ICWS), 2024, pp. 707–717.

[6] S. Al-Sarawi, N. Al-Saadi, M. Al-Hussein, and A. Ibrahim, "A quantum-assisted framework for energy-efficient cloud task scheduling using the quantum approximate optimization algorithm and variational quantum eigensolver," Preprint, 2025.

[7] S. K. Singh, R. Kumar, and P. Singh, "QRGEC: Quantum reinforcement learning with golden jackal optimization for resilient edge cloud coordination in internet computing," Sci. Rep., vol. 16, no. 1, pp. 1–15, 2026.

[8] V. Jha, S. Srivastava, A. Sharma, and R. Gupta, "Choreography and profiling of quantum-classical FaaS workflows on hybrid clouds," in Proc. 25th IEEE/ACM Int. Symp. Cluster, Cloud Internet Comput. (CCGrid), 2025.

[9] S. Giordano, K. Sen, and M. A. Martin-Delgado, "Leveraging cloud-native infrastructure for dynamic and flexible quantum-classical MLOps," Quantum Mach. Intell., vol. 8, no. 1, p. 8, 2026.

[10] M. Robbiati et al., "Qiboml: Towards the orchestration of quantum-classical machine learning," arXiv preprint arXiv:2510.11773, 2025.

[11] H. T. Nguyen, M. Usman, and R. Buyya, "DRLQ: A deep reinforcement learning-based task placement for quantum cloud computing," in Proc. 17th IEEE Int. Conf. Cloud Comput. (CLOUD), 2024, pp. 475–481.

[12] I. Singh, S. S. Gill, J. Sun, and J. Mol, "QAISim: A toolkit for modeling and simulation of AI in quantum cloud computing environments," Clust. Comput., vol. 29, no. 2, p. 99, 2026.

[13] H. T. Nguyen, M. Usman, and R. Buyya, "QFOR: A fidelity-aware orchestrator for quantum computing environments using deep reinforcement learning," arXiv preprint arXiv:2508.04974, 2025.

[14] P. J. Karalekas, N. A. Tezak, E. C. Peterson, C. A. Ryan, M. P. da Silva, and R. S. Smith, "A quantum-classical cloud platform optimized for variational hybrid algorithms," Quantum Sci. Technol., vol. 5, no. 2, p. 024003, 2020.

[15] F. Giust, V. Sciacca, and D. Scano, "Modeling and performance evaluation of hybrid classical–quantum serverless computing platforms," IEEE Trans. Quantum Eng., vol. 6, pp. 1–13, 2025.

[16] V. Stirbu, O. Kinanen, M. Haghparast, and T. Mikkonen, "Qubernetes: Towards a unified cloud-native execution platform for hybrid classic-quantum computing," Inf. Softw. Technol., vol. 175, p. 107501, 2024.

[17] E. Giortamis, A. Lenharth, K. S. McKinley, and S. T. King, "Orchestrating quantum cloud environments with Qonductor," arXiv preprint arXiv:2408.04312, 2024.

[18] E. Giusto et al., "Dependable classical-quantum computing systems engineering," Front. Comput. Sci., vol. 7, p. 1520903, 2025.

[19] R. Indhumathi et al., "Hybrid quantum–cloud framework for nonlinear clustering optimization and intelligent resource management," Results Nonlinear Anal., vol. 8, no. 3, pp. 26–35, 2025.

[20] A. Sharma and Z. Ali, "Integration of quantum computing with artificial intelligence: A systematic review," J. Eng. Res. Rep., vol. 27, no. 9, pp. 329–343, 2025.