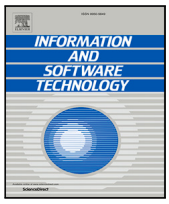
Information and Software Technology 191 (2026) 108005

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof



Maximizing quantum hardware utilization via multiprogramming circuits and shot-wise distribution

Giuseppe Bisicchia [a,b], Jaime Alvarado-Valiente [b,*], Javier Romero-Álvarez [b], Jose Garcia-Alonso [b], Juan M. Murillo [b], Antonio Brogi [a]

[a] Department of Computer Science, University of Pisa, Pisa, Italy
[b] Quercus Software Engineering Group, Universidad de Extremadura, Cáceres, Spain

ARTICLE INFO

Keywords:
Quantum computing
QSE
Multiprogramming
Shot-wise distribution

ABSTRACT

Context: Quantum computing is rapidly evolving, offering new opportunities for solving problems in optimization, cryptography, and simulation. However, the limited availability of quantum resources makes efficient utilization of quantum hardware a current challenge. Today's paradigms often lead to under-utilization of qubits, increased costs, and execution delays, especially in the NISQ era.
Objective: This work aims to improve the utilization of quantum hardware by introducing an execution model that integrates multiprogramming at circuit level with quantum shot-wise distribution in a single policy-driven pipeline.
Methods: An architecture has been implemented that combines circuit scheduling and shot distribution techniques to aggregate multiple circuits and distribute their shots across heterogeneous QPUs. The approach was empirically validated on actual IBM Quantum devices using a diverse set of reference circuits.
Results: The proposal achieved a reduction in cost of 95% and a reduction in tasks 92%. Moreover, the fidelity analysis of the results showed an increase in noise, with an average increase of approximately 20% using different statistical distances.
Conclusions: This research provides a usable and extensible solution to increase the efficiency, cost effectiveness, and resilience of quantum workload execution in heterogeneous and dynamic cloud environments. These results obtained suggest that users should weigh the implications of fidelity versus cost (and time) savings based on the application requirements and their goals.

1. Introduction

Recent years have witnessed remarkable progress in quantum hardware, with steadily increases in the number of available qubits, improvements in gate fidelity, and greater accessibility to quantum processors via cloud-based platforms [1]. However, despite this rapid evolution, the paradigms governing quantum circuit execution have not kept pace with hardware advancements. In virtually all currently available Quantum Processing Units (QPUs), the execution of each quantum circuit monopolizes the entire machine, even if the circuit utilizes only a small fraction of the available qubits [2]. This may lead to significant under-utilization of resources, especially in the Noisy Intermediate-Scale Quantum (NISQ) era, where the majority of practical circuits remain relatively small due to error rates, coherence limitations, and algorithmic constraints [3].

At the same time, the quantum ecosystem is increasingly characterized by heterogeneity and dynamism. Quantum hardware from different providers (e.g. IBM, Rigetti, IonQ, AWS Braket) varies not only in native gate sets and qubit connectivity, but also in queue dynamics, languages, and cost models [4]. This landscape is in constant flux, with individual devices exhibiting time-dependent variations in error rates and operational availability. Consequently, efficiently exploiting quantum hardware may benefit not only from maximized intra-device resource utilization, but also from adaptive strategies that leverage device heterogeneity for improved throughput, reduced cost, and higher reliability [5].

To address these challenges, the use of two methodologies is proposed. The first is *multiprogramming-based* [6], where multiple independent quantum circuits are scheduled and executed in parallel on a single QPU, maximizing hardware utilization and reducing wait times. In this way, the QCRAFT Scheduler [7,8] addresses the inefficiency of single-circuit execution by running multiple quantum tasks in parallel, but it can introduce new complexities in circuit mapping and potentially

* Corresponding author.
E-mail address: jaimeav@unex.es (J. Alvarado-Valiente).

https://doi.org/10.1016/j.infsof.2025.108005
Received 29 July 2025; Received in revised form 14 October 2025; Accepted 20 December 2025
Available online 20 December 2025

increase aggregate noise due to circuit concatenation and deeper circuit depths.

The second methodology is the *Shot-wise distribution* [9,10], a technique that decomposes the execution of quantum jobs at the shot level. Here, the shots required by a given circuit are distributed across multiple QPUs, possibly from different providers, allowing for parallel execution and effective exploitation of device heterogeneity. Shot-wise distribution can dynamically balance workload according to backend costs, queue state and noise characteristics, and is inherently robust to device failures. However, it comes at the expense of increased network traffic and potentially higher operational costs, as the number of submitted quantum jobs grows with fragmentation.

In this article, we propose combining these two complementary approaches in a synergistic process. By integrating circuit consolidation based on multiprogramming using circuit scheduling methods with Shot-wise distribution, we aim to capitalize on their respective strengths while compensating for their individual weaknesses. Scheduling methods maximize QPU utilization and drive down costs by combining circuits, while Shot-wise distribution leverages device heterogeneity and enhances resilience to backend failures or performance anomalies. Crucially, distributing jobs at the shot level increases fault tolerance, since failures or degraded performance on any single QPU affect only a fraction of the overall workload. Conversely, intelligent scheduling policies can reduce the increased traffic and overhead introduced by Shot-wise fragmentation.

The result is a flexible, policy-driven architecture capable of dynamically adapting to both the size and nature of quantum workloads, as well as to the evolving landscape of quantum hardware. In this work, we design, implement, and empirically evaluate such a hybrid pipeline, demonstrating substantial gains in cost efficiency, workload consolidation, and operational robustness, with controlled trade-offs in execution fidelity and system complexity.

This work extends our previous research on shot-wise distribution [9,10], and our previous research on circuit multiprogramming [7,8], by presenting, for the first time, a unified execution architecture that integrates multiprogramming with shot-wise execution. The novelty of this paper lies not in re-proposing these methods separately, but in their systematic combination, implementation, and empirical validation within a single modular framework. Specifically, our contributions are:

- **Unified Hybrid Pipeline:** We design and implement an integrated execution architecture that combines circuit-level multiprogramming with shot-wise distribution, enabling simultaneous consolidation of workloads and their distribution across heterogeneous QPUs.
- **Modular Architecture Components:** We introduce three core components: (i) the *Quantum Executor*, which decomposes shots and dispatches workloads; (ii) the *QCRAFT Scheduler*, which consolidates and multiprograms queued circuits using policy-driven strategies (e.g., AL-FIFO, replica-based scheduling); and (iii) the *Result Aggregator*, which merges distributed outputs into final result distributions.
- **Policy-driven Execution:** We provide flexible scheduling and allocation policies that allow the system to adapt dynamically to workload characteristics, backend costs, and error profiles.
- **Empirical Validation:** We conduct an extensive evaluation on real IBM quantum devices using benchmark circuits, showing that the proposed hybrid approach achieves up to 95% cost reduction and over 90% task reduction, while preserving acceptable fidelity levels in the NISQ era.

Our previous work introduced and formalized the *shot-wise* paradigm as a standalone methodology for quantum job execution. In the first paper [9], we motivated the idea of distributing the shots of a single circuit across heterogeneous QPUs and validated its feasibility, highlighting its benefits for workload balancing, robustness to backend variability, and reduction of queuing delays. In the subsequent, extended work [10], we developed a more general framework for shot-wise execution, introducing calibration- and reliability-aware policies, adaptive noise-aware split/merge strategies (e.g., based on Hellinger distance or MISE), and incremental execution mechanisms. That study provided a comprehensive analysis of how shot-wise policies influence fidelity and stability when distributing executions across multiple providers.

In contrast, this article does not re-propose shot-wise execution or circuit-level multiprogramming in isolation. Instead, it integrates shot-wise with multiprogramming into a single, unified pipeline, thereby addressing inefficiencies that arise when each methodology is applied separately. Specifically, while the earlier shot-wise papers demonstrated advantages at the *inter-device* level, they did not exploit intra-device parallelism through multiprogramming. Conversely, multiprogramming-only schedulers such as QCRAFT [7,8] improve hardware utilization but remain confined to a single backend and are vulnerable to device-specific noise and failures. By combining both strategies in a modular architecture, our work achieves cost reductions of up to 95%, workload consolidation above 90%, and improved robustness against backend fluctuations. This integration, supported by the new *Quantum Executor*, *QCRAFT Scheduler*, and *Result Aggregator* components, constitutes the central novelty and main contribution of this paper.

The remainder of this paper is organized as follows. Section 2 reviews the quantum scheduling and shot distribution methods used in this work. Section 3 describes our proposed architecture and its implementation. Section 4 presents the experimental evaluation and discusses the observed results. Section 5 outlines threats to validity. Section 6 reviews relevant related work, and Section 7 concludes with a summary of the findings and some directions for future work.

2. Background: Scheduling and distributing quantum circuits

The execution of quantum jobs on current NISQ hardware faces numerous limitations, including limited fidelity and long queue times on cloud-based quantum services. To address these challenges, two complementary strategies have emerged in the quantum software engineering domain, the aggregation of multiple circuits into shared executions and the distribution of circuit executions across multiple backends.

This section presents the two methodologies integrated in this work: the *Quantum Scheduler* and the *Shot-wise Distribution*.

2.1. Quantum scheduler

Quantum Scheduler is a service-based execution tool that enables the parallel queuing and joint execution of multiple quantum circuits by aggregating them into a single multiprogrammed circuit before executing in a QPU. This approach leverages the fact that most user-submitted circuits occupy only a small fraction of the available qubits on current QPUs [2]. By executing multiple circuits in parallel, it becomes possible to maximize qubit utilization, reduce queue times, and obtain lower execution costs [7,8].

Motivation and rationale

The increasing interest in quantum computing has led major providers such as IBM, Amazon, Google, and Microsoft to offer quantum resources through cloud platforms, following the Quantum Computing as a Service (QCaaS) model [11–13]. While the number of qubits in quantum processors is rapidly expanding, with IBM announcing devices up to 4000 qubits [14], current usage patterns indicate a significant under-utilization of resources. Recent studies show that, on average, only around 10 qubits are used per quantum circuit [2],

leading to inefficiencies such as high costs, long queue times, and limited access to machines [15].

In contrast, classical High-Performance Computing (HPC) environments mitigate similar problems through scheduling, load balancing, and resource orchestration techniques [16].

Methodology

The proposed scheduler is implemented as a service-oriented architecture that receives circuit submissions via an API. Users can submit circuits either as URLs from visual tools like Quirk or as Qiskit code, specifying the required number of shots. The system places these circuits in a queue and initiates execution cycles. During each cycle, it attempts to combine as many circuits as possible, without exceeding the qubit capacity of the target quantum processor using the *AL-FIFO (At Least First-In, First-Out) policy*. This policy ensures that, at minimum, the circuit at the head of the queue will be included in each scheduling cycle. In addition, it offers another policy, the *Replica-based scheduler policy*, which schedules a circuit with itself, in order to reduce the number of shots and maximize machine utilization.

Combined circuits are submitted to IBM Quantum processors before transpilation, which ensures that qubit mapping is adjusted without overlaps. After execution, an unscheduling process extracts individual results based on the qubits assigned to each circuit. Developers can retrieve their results by referencing the execution ID provided at submission.

When to use quantum scheduler

This scheduler is particularly useful in environments where multiple circuits of relatively low qubit count need to be executed but are constrained by long queue times and high costs. Developers working with Hybrid Classical-Quantum systems or prototyping quantum services can benefit from reduced overhead, especially during the NISQ era [17], when frequent testing is required and qubit resources are expensive. Moreover, the Scheduler is valuable when the goal is to maximize the utilization of available hardware, such as IBM's 127 or 156 qubit processors.

Applications and extensions

Potential applications of the scheduler include batch processing of quantum algorithms, optimization of resource usage in multi-user environments, and enabling real-time hybrid systems where execution latency must be minimized.

Furthermore, the tool can be extended to support dynamic shot allocation, integration with circuit editing while in queue, or policies prioritizing certain tasks.

Limitations and challenges

Despite its benefits, the scheduler must operate within the constraints of NISQ hardware, where noise, decoherence, and limited fidelity can affect results, especially for deep or entangled circuits [18]. Moreover, although transpilation avoids qubit conflicts, certain circuit combinations may exacerbate gate errors or depth, leading to noise constraints. Statistical validation has shown that these effects are minor in most cases, but some circuits exhibit higher sensitivity.

Additionally, the scheduler has been tested primarily on IBM hardware, which may limit generalization to other quantum computing platforms with different architectures or queueing policies.

2.2. Shot-wise distribution

Shot-wise distribution is a recently introduced quantum execution methodology that enables the *parallel execution* of (even) a single quantum circuit by distributing individual measurement shots across multiple QPUs [9,19]. This technique leverages a fundamental property of quantum computing: Each measurement shot yields an independent sample from the circuit's output probability distribution. Consequently, it is feasible to split the total number of shots among different devices without compromising the statistical validity of the final results.

Motivation and rationale

The NISQ era is defined by resource constraints such as limited qubit counts, short coherence times, low gate fidelity, and constrained device access due to queuing and limited throughput. Moreover, most practical quantum algorithms require a large number of shots to produce reliable and statistically significant outcomes.

In traditional execution models, all shots are submitted to a single QPU, which can lead to long wait times, bottlenecks, and increased susceptibility to device-specific failure. The Shot-wise distribution mitigates these challenges by partitioning the execution across multiple QPUs (potentially from different quantum and cloud providers), thereby reducing latency, increasing execution reliability, and improving overall throughput. Importantly, recent experimental evidence highlighted that Shot-wise distribution can enhance the stability and reliability of results by mitigating device-specific noise and variability while reducing worst-case outcomes [10].

In cloud-native and multi-user environments, the heterogeneity of available devices in terms of performance, cost, and queue depth presents an opportunity for intelligent workload management. The Shot-wise distribution enables dynamic, fine-grained load balancing and cost optimization by assigning shot workloads based on real-time system metrics [9].

Methodology

The Shot-wise distribution pipeline is typically composed of the following four stages:

- **Device Selection:** Identify a pool of available QPUs based on criteria such as gate and readout fidelity, qubit topology, queue status, geographic distribution, and usage pricing. This stage sets the foundation for optimized multi-device execution.
- **Shot Allocation:** Distribute the total shot budget among the selected devices. Allocation strategies can be uniform, probabilistic, or adaptive. Adaptive approaches leverage real-time telemetry (e.g., queue length, runtime errors, device uptime) to dynamically reassign shot loads, improving efficiency and robustness.
- **Parallel Execution:** Once assigned, the individual shot batches are executed in parallel across the selected QPUs. This parallelism may shorten total execution time, particularly when devices are subject to different scheduling or queuing constraints.
- **Result Aggregation:** After execution, the individual shot results are merged to reconstruct the circuit's final output distribution. This aggregation is typically performed via weighted averaging, but more sophisticated statistical inference techniques can be employed to account for fidelity variation and shot count imbalances across devices.

When to use shot-wise distribution

Shot-wise distribution is designed to offer advantages in any of the following scenarios:

- **Multi-device access:** When multiple QPU are accessible.
- **Time-sensitive execution:** When reducing runtime or avoiding queuing delays is critical.
- **Cost-awareness:** When optimizing resource expenditure is a priority, especially in usage-based billing models.
- **Fault tolerance:** When robustness to single-device failure is required.

Importantly, the classical computational overhead incurred by splitting and aggregating shots is usually negligible relative to the quantum execution time [20]. As such, the benefits of parallelization and reliability typically outweigh coordination costs.

Applications and extensions

Shot-wise distribution has already been integrated with *circuit cutting* [21,22] methodologies, where a large circuit is partitioned into smaller, executable fragments. Within such pipelines, each fragment's shots can be independently distributed, enabling scalable execution across heterogeneous backends [20].

Additionally, noise-aware shot-wise policies, that address the challenge of heterogeneous device error profiles, are under active investigation. In [10], several advanced split-merge strategies were proposed, including Hellinger-distance and *Mean Integrated Square Error* (MISE) based weighting, combined with calibration phases to rank devices by reliability. Experimental evidence showed that, while such strategies do not outperform the single best QPU, they consistently improved worst-case performance, reduced variability, and provided more stable aggregated results across heterogeneous backends. These results highlight how statistical techniques can indeed mitigate the impact of device-to-device noise differences. However, in the present work we deliberately restricted ourselves to the simplest uniform allocation and naive aggregation strategy. This choice was made to isolate and evaluate the fundamental feasibility and benefits of the shot-wise distribution paradigm without confounding factors introduced by more sophisticated policies.

Limitations and challenges

The primary shortcoming related to Shot-wise distribution is that, by executing portions of the same workload on multiple devices, the total number of quantum jobs submitted to the ecosystem increases. This leads to a larger quantum workload footprint and potentially amplifies network traffic, queue congestion, and scheduling complexity across quantum cloud infrastructures.

In high-demand environments, elevated system traffic may offset some of the time-saving benefits by exacerbating contention for limited quantum resources. Furthermore, from a provider perspective, the increased fragmentation of workloads could lead to inefficiencies in backend utilization.

3. Proposal, architecture, and implementation

This section details our proposal: a unified, modular pipeline that combines Shot-wise distribution and advanced quantum scheduling for efficient multi-programming in quantum computing environments. Our system is instantiated in the QCRAFT Scheduler-Shotwise framework[1] and is designed to maximize QPU utilization, reduce costs, and flexibly support heterogeneous quantum backends.

Our objective is to develop a pipeline that:

1. Accepts quantum workloads from multiple developers as arbitrary quantum circuits, each with user-defined shot counts;
2. Efficiently partitions and distributes these workloads across heterogeneous QPUs (both local and remote, from multiple providers);
3. Employs dynamic, policy-driven scheduling to maximize resource utilization, minimize execution time, and reduce costs;
4. Transparently merges results and provides developers with final output distributions corresponding to their initial job submissions.

3.1. System architecture overview

The proposed architecture, shown in Fig. 1, consists of three principal components: (1) the *Quantum Executor*, (2) the *QCRAFT Scheduler*, and (3) the *Result Aggregator*.

[1] Open source and freely available at: https://github.com/Qcraft-UEx/QCRAFT-Scheduler-Shotwise.

1. Quantum executor

The **Quantum Executor** [23] serves as the primary orchestrator for the incoming quantum workloads. It is responsible for:

- Receiving developer-submitted quantum jobs: quantum circuit and required number of shots;
- Decomposing each job by splitting the required shots among available QPUs using configurable shot allocation policies (e.g., uniform, proportional, cost-aware, noise-aware [10]);
- Generating a *dispatch plan* that maps each quantum circuit to one or more QPUs, along with a designated subset of the total shots.

This process enables fine-grained parallelism and improves load balancing, leveraging the independence of measurement shots in quantum circuits.

2. QCRAFT scheduler

The **QCRAFT Scheduler** [7] acts as a queue manager and workload consolidator for each QPU. Its key features include:

- **Queuing Circuits:** Incoming circuits (possibly from different developers and jobs) are placed into a queue specific to each backend.
- **Circuit multiprogramming:** The scheduler applies user-selectable policies to combine multiple circuits into a single *envelope circuit* whenever backend resources (e.g., qubits available) allow. This multiplexing maximizes qubit usage and reduces backend idle times.
- **Policy Control:** The scheduler supports two scheduling policies that guide how envelope circuits are constructed. The default is AL-FIFO (At Least First-In First-Out), a variation of the traditional FIFO approach. In AL-FIFO, the scheduler guarantees that the first circuit in the queue is always selected for execution, and, if resources allow, additional circuits are appended without delaying the leading job. Another available policy is the *Replica-based scheduler policy*. It is a built-in execution strategy within the QCRAFT Scheduler that enables automatic circuit replication to maximize qubit utilization on a target quantum backend. Thanks to their modularity, these policies can be applied individually or in conjunction. Furthermore, they can be substituted with new policies as they are implemented, by adding them as new endpoints within the API.

Upon execution completion, the scheduler unschedules (demultiplexes) the combined results and forwards the individual output distributions for each developer circuit to the Result Aggregator.

3. Result aggregator

The **Result Aggregator** (a component of the Quantum Executor) is responsible for merging the shot-wise results. For each original developer job, it:

- Collects all partial (demultiplexed) results returned from distributed QPU executions;
- Aggregates them according to a user-defined merge policy (using weighted averaging or more advanced, fidelity-aware statistical techniques if desired);
- Produces a final output distribution per job, preserving the statistical validity of the original quantum experiment.

3.2. Pipeline workflow

The end-to-end workflow of our system, also shown in Fig. 1, is described below:

1. **Job Submission:** Developers submit quantum circuits specifying required shots and backends. Submissions are received by the Quantum Executor via API.

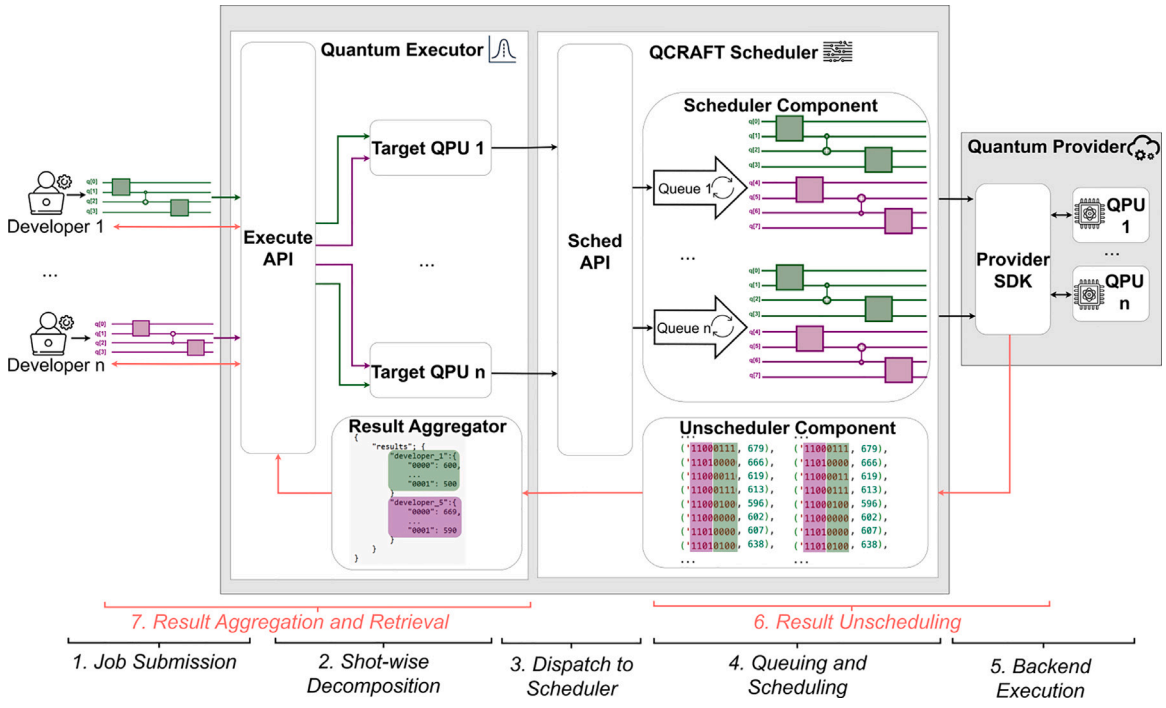


Fig. 1. Main Architecture.

2. **Shot-wise Decomposition:** For each job, the Quantum Executor splits the shot count among the available QPUs, based on the configured shot allocation policy. Each QPU is assigned one or more partial jobs.
3. **Dispatch to Scheduler:** The Quantum Executor dispatches these jobs to the QCRAFT Scheduler associated with each target backend.
4. **Queuing and Scheduling:** The Scheduler maintains per-backend queues. According to the selected scheduling policy, it combines multiple circuits into a single job (compatibly with qubit availability) and multiprograms this *envelope* for execution on the backend.
5. **Backend Execution:** Each QPU transpiles and executes the assigned envelope circuits and returns the results of measuring the final states of the qubits.
6. **Result Unscheduling:** The output (measurement results) is demultiplexed into separate results for each circuit and shot batch. Therefore, the obtained partial results are a combination of all the results of each individual circuit.
7. **Result Aggregation and Retrieval:** All partial results are passed to the Quantum Executor, where the Result Aggregator aggregates them using a merge policy. The developer receives a single and merged output distribution per circuit.

3.3. Architectural choices

The architecture is intentionally modular, separating concerns between the shot-wise partitioning logic (Quantum Executor) and backend resource management (QCRAFT Scheduler). This separation enables:

- **Policy Agility:** Each layer can independently design and evolve its policies (e.g., introducing new scheduling algorithms or adaptive shot-splitting strategies).
- **Provider Abstraction:** The system seamlessly supports multiple backend types and cloud providers, future-proofing the architecture for the evolving quantum landscape.
- **Scalability and Robustness:** Decoupling the executor and scheduler allows for distributed deployment and horizontal scaling.

Our QCRAFT Scheduler-Shotwise architecture provides a usable, extensible, and efficient pipeline for orchestrating quantum workloads in the NISQ era. By integrating shot-wise distribution with backend-aware scheduling and result aggregation, we aim at cost reductions, improved resource utilization, and robust support for heterogeneous, multi-tenant quantum computing environments.

4. Evaluation & results

To evaluate the impact of integrating the Shot-wise distribution methodology with scheduling techniques, we conducted a comprehensive experimental study using 13 benchmark quantum circuits from the MQT Bench suite [24]. These circuits cover a broad spectrum of quantum computing domains and were categorized as follows: **combinational circuits** (Simon — 6 qubits, Quantum Fourier Transform (QFT) — 3 qubits, Bernstein-Vazirani — 4 qubits, Shor — 4 qubits, Full Adder — 4 qubits, Deutsch-Jozsa — 4 qubits, Grover — 9 qubits, Kickback — 3 qubits), **sequential circuits** (Quantum Walk — 5 qubits, Phase Estimation — 6 qubits), and **variational circuits** (QAOA — 6 qubits, Traveling Salesman Problem (TSP) — 5 qubits). This selection features diverse structures in terms of door depth, connectivity, and measurement operations.

These circuits use from 3 to 9 qubits, with an average width of approximately 5 qubits. It aligns with current usage trends in the NISQ era, where most practical circuits remain below 10 qubits due to hardware noise and connectivity constraints [2].

A total of three experiments were performed. In each experiment, all 13 circuits were scheduled into a single task, alongside the baseline execution of each circuit individually (traditional execution). For every experiment, we used 10,000 measurement shots as the baseline. When applying the Shot-wise methodology across two QPUs, the shots were divided equally, allocating 5000 shots per QPU.

- Bare Shot-wise distribution without scheduling.
- Shot-wise + Scheduler: combination with circuit-level scheduling using the AL-FIFO policy.

- Shot-wise + Scheduler + Replica-based: full pipeline, including automatically circuit replication policy.

Shot-wise was configured with *uniform* split and merge policies (i.e., all backends were weighted equally in both policies [10]). Executions were carried out on two IBM Quantum devices with different topologies and capacities: *ibm_brisbane* and *ibm_sherbrooke*. This enabled an analysis of the workload distribution across different QPU topologies.

4.1. Results validation

To assess the impact and noise of each execution methodology on the fidelity of the outcome, different statistical distances were used [25], specifically the Hellinger distance, the Total Variation Distance (TVD), and the Jensen–Shannon divergence (JSD). These metrics are commonly employed in the analysis of quantum noise and distance estimation in NISQ circuits, as outlined by Sazzad et al. [26], due to their interpretability and robustness in measuring distributional divergence in probabilistic quantum outputs. Specifically, in this work, these statistical distances measure the deviation of the probability distributions resulting from the sequential execution of the circuits and the results obtained after executing the different proposed methodologies.

The Hellinger distance is a symmetric, bounded measure particularly sensitive to small deviations and well-suited for capturing probabilistic fidelity loss. The TVD represents the maximum difference between two distributions over all possible outcomes. The JSD provides a symmetric divergence metric based on the Kullback–Leibler divergence. Unlike TVD or Hellinger, the JSD has the advantage of being always finite and normalized, making it easier to interpret when comparing noisy probability distributions that may arise from heterogeneous sources of errors (e.g., gate noise or decoherence).

Therefore, the three distances compare the probability distribution obtained from a given execution method with a reference distribution produced by an ideal sequential execution. They return values between 0 and 1, where 0 indicates perfect similarity, and higher values signal increasing divergence.

In addition to this methodological validation, we also performed baseline comparisons against the standard execution model in which all shots are executed on a single quantum device without any form of scheduling or shot distribution. This baseline represents the typical execution strategy available to end-users on current cloud-based QPUs. Comparing against this reference highlights that our Shot-wise methodology maintains a very similar level of fidelity, while introducing the flexibility of multi-device execution. This indicates that the fidelity degradation observed in subsequent strategies (Scheduler and Replica-based) is primarily attributable to the multiprogramming-induced circuit depth increase, rather than the fundamental act of distributing shots.

Furthermore, it is important to note that the experiments were executed on two IBM Quantum devices with distinct topologies and qubit capacities, namely *ibm_brisbane* and *ibm_sherbrooke*. These devices exhibit diverse noise profiles, as IBM hardware is recalibrated daily and error rates vary both across qubits within a device and between devices. In practice, this means that fidelity is influenced not only by the execution methodology but also by backend-specific noise characteristics. The Shot-wise distribution benefits from this heterogeneity by averaging results across devices, which can sometimes mitigate device-specific error spikes. Conversely, when multiprogramming is applied, the deeper composite circuits amplify the effects of noise on whichever device is chosen, making the impact of backend calibration more pronounced. Therefore, while our results show an average 20–22.5% fidelity degradation for the multiprogrammed strategies, this value should be interpreted in the context of heterogeneous device error landscapes, where the balance between cost savings and fidelity preservation depends on the interplay of both execution strategy and device noise.

Across all three metrics, the results shown in Fig. 2 reflect that the basic Shot-wise configuration consistently achieves low divergence values, usually below 0.1. That indicates high fidelity across circuits. When the Scheduler is incorporated, the divergence increases as expected due to the parallel execution of the circuits and the resulting increase in depth. The average increase in Hellinger is 22.4%, in TVD is approximately 20%, while JSD shows an average increase of 5% compared to the baseline.

Finally, adding the Replica-based scheduler policy introduces a similar level of loss of fidelity (22.5% on average).

4.2. Cost and task reduction analysis

Fig. 3 presents the cost in USD for each circuit and configuration.

As shown, individual execution reaches a total cost of $76.8, while the Shot-wise alone incurs a significantly higher cost of $134.4, a 75% increase due to job fragmentation and duplicated overhead across QPUs. By incorporating the Scheduler, the cost drops sharply to $6.73, achieving a 91.24% cost reduction, while also reducing the number of tasks by over 92%, going from executing 12 tasks to 2.

Finally, the incorporation of the Replica-based scheduler policy (Shot-wise + Scheduler + Replica-based) results in the lowest execution cost, only $3.74, which represents a 95.13% cost reduction relative to the individual baseline and an 88.31% improvement over the basic Shot-wise methodology. This confirms that the integration of circuit-level multiprogramming into a distributed execution pipeline leads to significant cost savings and enhanced workload consolidation across QPUs.

Overall, although circuit programming methodologies introduce moderate degradation in fidelity, these deviations remain within acceptable margins in the NISQ era and are balanced by reductions in execution cost and task overhead. Therefore, the choice between higher fidelity or higher cost reduction with the use of the proposed methodologies is left to the developers.

5. Threats to validity

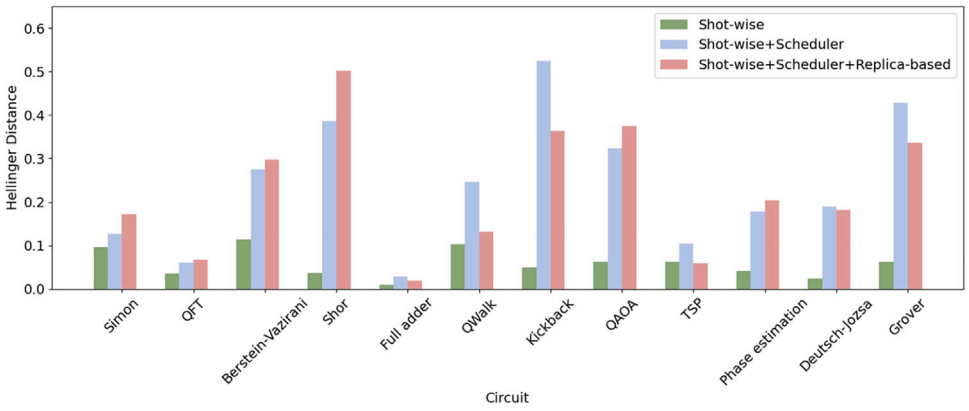
In this section, we discuss potential threats to the validity of our study. While we have taken rigorous steps to ensure the reliability and robustness of our results and open-sourced our tool, codebase, and experimental data, certain limitations inherent to quantum computing environments and experimental design must be acknowledged.

5.1. Internal validity

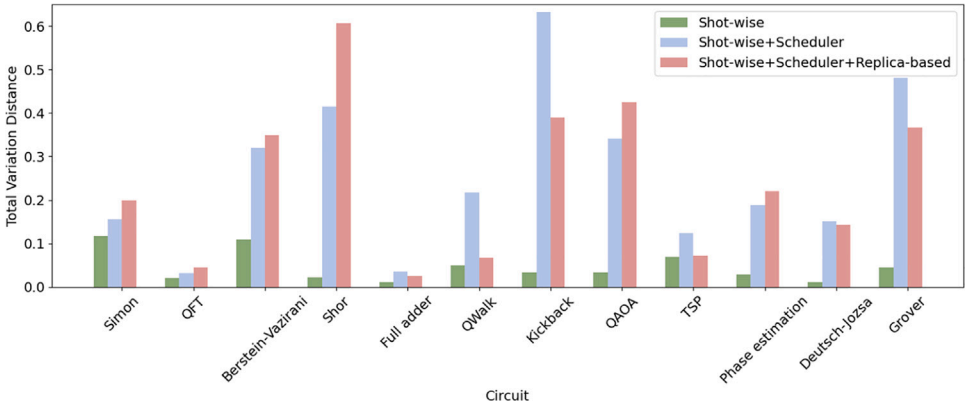
Internal validity concerns the correctness of the implementation and the interpretation of causal relationships in our study. A key source of internal threat is the dynamic calibration and error profile of real QPUs. Quantum devices, especially those accessible through cloud providers, undergo periodic recalibration, which alters gate fidelities, decoherence times, and readout errors. Since our experiments span multiple executions over time, minor fluctuations in hardware performance could have influenced the observed differences in cost and fidelity.

To mitigate this, we repeated all benchmark executions within controlled temporal windows, ensuring that device calibrations were stable during the tests. Nonetheless, some residual variability may persist.

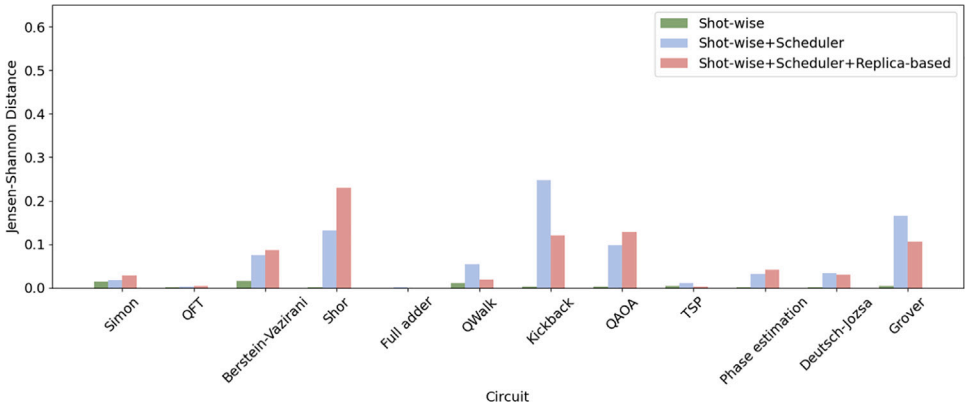
Additionally, there exists the well-known problem of crosstalk in circuit multiprogramming. Crosstalk refers to the undesired interference between qubits or gates belonging to different circuits that are executed simultaneously on the same QPU [27]. We acknowledge that our current tool does not directly address inter-circuit crosstalk. However, in our empirical validation, we compare the fidelity of the results from the multiprogrammed execution against the baseline of single-circuit execution. This comparison allows users to decide if the cost and



(a) Hellinger Distance



(b) Total Variation Distance (TVD)



(c) Jensen–Shannon Divergence (JSD)

Fig. 2. Noise validation results for the 12 benchmark circuits using three statistical metrics: (a) Hellinger Distance, (b) Total Variation Distance (TVD), and (c) Jensen–Shannon Divergence (JSD). Green bars represent the Shot-wise methodology, blue bars the combination of Shot-wise and Scheduler, and red bars the complete pipeline including the Replica-based Scheduler policy.

task reduction benefits of our approach are sufficient to compensate for the fidelity degradation. For those applications requiring explicit mitigation, in other works we propose alternatives to try to mitigate the increased crosstalk caused by the scheduling. Specifically, we propose the utilization of quantum islands [28] within the QCRAFT Scheduler to reduce crosstalk between qubits by isolating or separating the qubits used by different circuits on the QPU topology. A quantum island in circuit multiprogramming refers to a set of qubits used by a circuit that are isolated or separated from those used by other circuits by one or more qubits within the same topology.

5.2. External validity

External validity refers to the generalizability of our findings to other settings or environments. Our study was conducted using a selected set of quantum circuits (MQT Bench), a limited number of real QPUs from IBM, and specific scheduling and distribution policies. While the circuits represent a diverse range of domains and structures, they may not encompass all possible workload types or quantum algorithm behaviors.

Furthermore, although the QCRAFT Scheduler and Quantum Executor support multiple providers, our evaluation was restricted to IBM

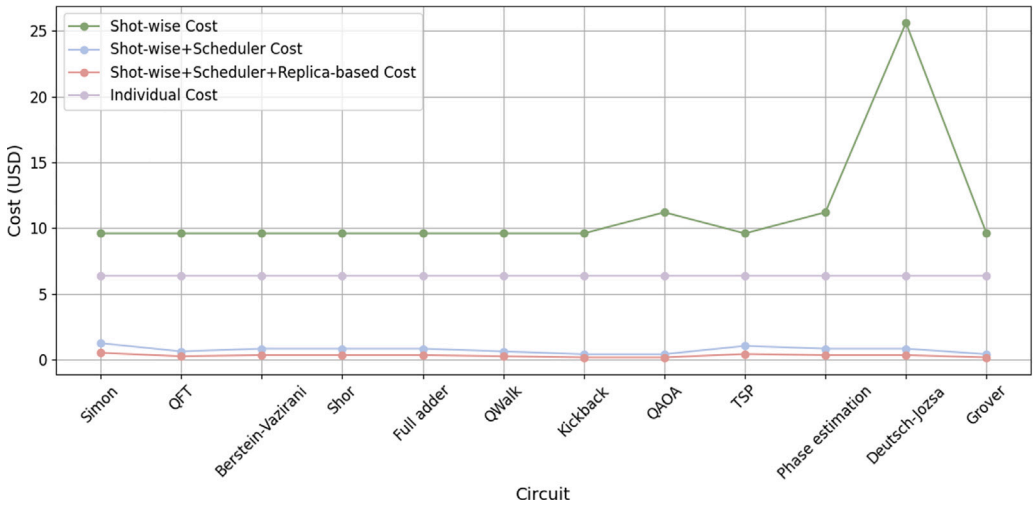


Fig. 3. Comparison of costs between different methodologies. The green line represents the baseline cost using only the Shot-wise, while the purple line indicates the cost of executing each circuit individually. The blue and red lines correspond to the addition of the Scheduler and Replica-based scheduler policy, respectively.

QPUs due to availability and access constraints. Therefore, performance and reliability outcomes may vary when applying our pipeline to other hardware ecosystems such as IonQ, Rigetti, or OQC. Future work should aim to replicate this study across different hardware stacks to enhance generalizability.

5.3. Construct validity

Construct validity relates to how well our metrics and methods reflect the intended theoretical constructs. In our evaluation, we used cost (measured in USD) and fidelity (measured using the Hellinger Distance, TVD, and JSD) as primary metrics. While these are well-established indicators, they may not fully capture qualitative factors such as user-perceived reliability, usability of tools, or error propagation in downstream quantum applications.

Therefore, complementary metrics such as KL divergence or fidelity-specific cost functions could be considered in future analyses.

5.4. Conclusion validity

Conclusion validity addresses the statistical soundness of the analysis and whether the reported effects are robust and significant. While we report substantial cost and task reduction benefits from combining scheduling and shot-wise strategies, we did not apply hypothesis testing or confidence interval estimation, since our goal was to perform an empirical demonstration rather than a hypothesis-driven study. Future work should incorporate statistical tests to further validate the observed patterns.

We released all source code, configurations, and benchmark scripts to facilitate verification and replication by the community.

6. Related work

Quantum workload orchestration and efficient resource utilization are growing areas of interest in the domain of Quantum Software Engineering, especially in the context of the NISQ era. Broadly, prior research in this space can be categorized into two complementary branches: (i) circuit-level scheduling techniques that aim to optimize quantum workload execution on single devices and QPU resource utilization; and (ii) distribution methodologies that seek to exploit backend heterogeneity by executing workloads across multiple quantum processors.

While scheduling strategies have focused on backend-aware resource consolidation, often optimizing qubit usage or minimizing latency, distribution techniques aim to distribute and/or decompose quantum jobs across diverse devices to improve throughput, reduce queuing delays, and increase fault tolerance. In this section, we survey representative contributions from both branches and compare them with the integrated methodology developed in this work.

6.1. Maximizing resource utilization

Quantum Software Engineering is emerging as a key discipline to bridge classical software practices with quantum computing capabilities [29–31]. Among its core paradigms is Hybrid Classical-Quantum computing, which integrates classical computing with quantum execution to enhance scalability and performance [32]. This paradigm has facilitated the rise of quantum services, allowing developers to interact with quantum hardware via a service-oriented model [29].

As quantum computing gains adoption, resource management becomes increasingly critical. The limited availability of QPUs, combined with inefficient infrastructures, often results in long queuing delays ranging from seconds to hours [33]. Despite the rapid growth in qubit capacity, actual usage remains low, averaging around 10 qubits per circuit [2], leading to under-utilized hardware and high execution costs. These issues are expected to persist as quantum hardware remains primarily available through cloud platforms [34].

While other works have explored circuit-level optimizations, such as gate count reduction [35,36] or depth minimization using deep learning [37], these works focus on individual circuit fidelity. Other efforts address infrastructure-level optimization. For example, *Delayed Instruction Scheduling (DIS) policy* [38] enables concurrent circuit execution through qubit partitioning to support multiprogramming; *QuCloud+* [3] introduces a system for efficient qubit mapping and dynamic resource allocation in multi-application environments; and *Online Job Scheduler* [6] also explores multiprogramming quantum jobs in fault-tolerant quantum computers. In addition, *QGroup* [39] proposes a parallel quantum job scheduling approach based on dynamic programming. By grouping jobs and optimizing their parallel execution, QGroup improves throughput and resource utilization across QPUs.

Finally, *SCIM* [40], along with the circuit cutting strategy proposed in [41], explores workload parallelization by decomposing circuits or distributing their execution across quantum resources.

In contrast, the QCRAFT Scheduler emphasizes the scheduling of multiple independent quantum circuits into a single execution by multiprogramming. This aligns with the service-oriented nature of quantum applications and aims to improve hardware efficiency without modifying the internal circuit structure or introducing slicing complexity.

6.2. Distributing quantum computations

The problem of selecting the most appropriate QPUs for executing a quantum circuit has received significant attention in the literature. Most existing approaches address QPU selection by defining and optimizing a set of metrics that may include not only the physical and logical characteristics of available quantum devices (such as qubit count, topology, or noise model), but also circuit-specific features (gate count, width, depth), and dynamic environmental factors such as queue waiting time, cost, and backend availability.

A prominent example is *Quantum API Gateway* [42], which provides a service for automatic selection of an optimal QPU for each submitted quantum circuit. The selection process considers factors such as device architecture (e.g. gate-based vs. annealing) and circuit width, and enables users to prioritize speed or cost according to their application requirements.

The *NISQ Analyzer* [43] follows a similar strategy, but also considers that multiple circuit implementations may exist for a single quantum algorithm. It selects the best circuit implementation from a repository of candidates using selection rules dependent on input data, and then determines the most suitable QPU-circuit pair according to criteria such as circuit width, depth, and SDK compatibility. The NISQ Analyzer ecosystem has been extended to include tools for compiler comparison [44], ML-based optimization for discarding non-promising compilers and QPUs prior to compilation [45], and further compilation optimization using machine learning [46].

Beyond these, several other approaches have investigated intelligent QPU selection and workload management. For example, [47] introduces an adaptive job scheduler that selects QPUs by jointly considering estimated output fidelity and queue waiting time. The framework in [48] addresses the integration of quantum computing within enterprise cloud systems, using static attributes such as qubit count and queue length to select a suitable device. In a similar vein, [49] presents a system for automatically predicting the best combination of QPUs, compilers, and compilation parameters to maximize the execution fidelity for a given quantum circuit.

Addressing the problem from another perspective, the *EQC framework* [50] proposes *ensembled quantum computing* for Variational Quantum Algorithms. Instead of relying on a single device, EQC dynamically distributes different circuit executions across a heterogeneous set of QPUs, asynchronously managing noise-aware weighting and backend selection. This ensemble-based approach reduces device-specific bias, mitigates calibration drift, and significantly accelerates training by parallelizing workload distribution across multiple machines. In contrast to our *shot-wise distribution* methodology, which distributes measurement shots of the *same circuit instance* across QPUs to exploit statistical independence and enable fine-grained parallelism, EQC focuses on distributing *entire circuit evaluations* (e.g., in variational optimization) across devices. Thus, while EQC improves the efficiency and robustness of iterative training loops, shot-wise distribution enables resilience and scalability at the level of measurement sampling, targeting complementary aspects of distributed quantum computation.

More recently, Li et al. introduced *QuSplit* [51], a framework that addresses the fidelity-throughput tradeoff by splitting quantum jobs across noisy devices. This work improves both execution fidelity and throughput by partitioning jobs into smaller sub-jobs that can be run concurrently, thereby mitigating the effect of noise accumulation and queue delays. Compared to our proposal, QuSplit highlights the advantages of job partitioning strategies. While both approaches share the objective of improving efficiency and reliability on NISQ devices, our method enables fine-grained control at the level of measurement shots and circuit scheduling, whereas QuSplit operates at the job level to balance fidelity and throughput.

Despite these advances, existing proposals predominantly focus on identifying a single best QPU (or QPU-circuit pair) for the entire quantum workload, and then executing all measurement shots on that device. In contrast, our approach departs from this paradigm by leveraging the statistical independence of quantum measurement shots through the *shot-wise distribution* methodology [9,10], wherein the shots (even those from a single circuit) are distributed across multiple, heterogeneous QPUs, thereby enabling parallelism, increased resilience, and adaptive load balancing.

Final remarks. To the best of our knowledge, this work presents the first framework that combines circuit-level scheduling and shot-wise distribution into a unified quantum workload execution pipeline. While prior studies have investigated these two strategies in isolation, optimizing either intra-device resource utilization or inter-device distribution, no existing approach integrates them in a modular and policy-driven architecture.

7. Conclusion

In this paper, we present a hybrid execution model for quantum computing that integrates two complementary paradigms: *circuit-level scheduling* and *shot-wise distribution*. Our approach is motivated by the limitations of current quantum cloud infrastructures, where resource under-utilization, device heterogeneity, and lack of fault tolerance remain key barriers to efficiency and scalability. By combining the fine-grained load distribution benefits of shot-wise execution with the hardware utilization optimizations offered by multi-programmed scheduling, we deliver a flexible pipeline for quantum job orchestration.

We implemented this model in the QCRAFT Scheduler-Shotwise framework and evaluated its effectiveness using 12 benchmark circuits from the MQT Bench suite. The experiments covered execution on real IBM QPUs and assessed three main configurations: pure Shot-wise execution, Shot-wise with scheduling, and Shot-wise with both scheduling and making use of the Replica-based scheduler policy. Our evaluation demonstrates the following key findings:

- **Cost Reduction:** The integrated approach achieved a total cost reduction of up to 95.13% compared to sequential execution and 88.31% compared to naive shot-wise execution alone.
- **Task Reduction:** The number of quantum jobs submitted to the cloud was reduced by more than 92% with the full pipeline, reducing system pressure and queue congestion.
- **Fidelity Impact:** The inclusion of circuit-level scheduling introduced an average increase of approximately 22.5% in the Hellinger Distance 20% in TVD, and 5% in JSD, mainly due to deeper composite circuits. These results suggest that developers must weigh the implications of fidelity versus cost savings based on the specific requirements and tolerance levels.
- **Fault Tolerance and Resilience:** Shot-wise distribution improved robustness by spreading the execution across multiple devices, mitigating the impact of backend-specific errors or failures.

Overall, the proposed architecture provides a practical and extensible solution to increase the efficiency, cost-effectiveness, and resilience of quantum workload execution in heterogeneous and dynamic cloud environments. Our results confirm that the trade-offs introduced by each technique can be mutually balanced when combined into a unified system, leading to a more favorable operational profile.

It is worth noting that the benefits of load balancing not only persist but may become even more significant when extending execution beyond a single provider. By distributing workloads across heterogeneous QPUs, users are no longer constrained by the limitations of a single provider. Different providers expose diverse hardware characteristics, including qubit technologies and counts, connectivity topologies, gate fidelities, queue dynamics, access modalities, and cost models. Load balancing empowers the orchestration layer to select the most suitable combination of hardware for each circuit, aligning backend features

with application requirements. This flexibility can enhance execution quality while also ensuring more predictable costs and turnaround times. As we previously discussed in [9], shot-wise load balancing across heterogeneous QPUs can improve adherence to Quality of Service (QoS) requirements, enabling more effective handling of scenarios with stringent constraints and multi-objective optimization functions. Moreover, recent experimental results reported in [10] highlight that shot-wise distribution can also be leveraged as a form of noise mitigation when allocating shots across multi-provider QPUs. By exploiting backend diversity in error rates and calibration profiles, the aggregated results can yield more accurate output distributions than relying on a single noisy device. This evidence further reinforces that load balancing across heterogeneous hardware not only addresses cost and performance trade-offs but also contributes to improving fidelity and robustness of quantum executions.

Future work

There are several promising avenues for future research and enhancement:

Adaptive Scheduling and Shot Allocation Integrating machine learning models to predict backend queue times, error rates, and pricing dynamics could enable more intelligent, real-time decision-making in both scheduling and shot-splitting strategies.

Fidelity-Aware Aggregation More advanced statistical techniques could be used to weight partial results during Shot-wise merging based on backend-specific fidelity profiles, leading to higher overall result accuracy.

Multi-provider Evaluation Future experiments will incorporate additional quantum hardware providers (e.g., IonQ, Rigetti, AWS Braket) to assess cross-platform behavior and refine policies for highly heterogeneous ecosystems.

Security and Provenance As execution pipelines become more distributed, guaranteeing integrity, reproducibility, and security of quantum job data across multiple providers is a critical area for exploration.

CRediT authorship contribution statement

Giuseppe Bisicchia: Writing – original draft, Validation, Software, Investigation, Conceptualization. **Jaime Alvarado-Valiente:** Writing – original draft, Validation, Software, Investigation, Conceptualization. **Javier Romero-Álvarez:** Writing – original draft, Validation, Investigation, Conceptualization. **Jose Garcia-Alonso:** Writing – review & editing, Visualization, Supervision, Methodology. **Juan M. Murillo:** Writing – review & editing, Supervision, Methodology, Funding acquisition. **Antonio Brogi:** Writing – review & editing, Supervision, Methodology, Funding acquisition.

Funding

This work has been partially funded by the European Union "Next GenerationEU/PRTR", by the Ministry of Science, Innovation and Universities (TED20 21-130913 B-I00, PDC2022-133465-I00). It is also supported by QSERV project (PID2021-1240454OB-C31) and ATHENA project (PID2024-155693NB-C41) funded by the Spanish Ministry of Science and Innovation and ERDF; by European Union under the Agreement — 101083667 of the Project "TECH4E -Tech4effiencyEDlH" Call: DIGITAL-2021-EDlH-01 supported by the European Commission through the Digital Europe Program; by the Regional Ministry of Education, Science and Vocational Training of the Regional Government of Extremadura (GR24099). Grant PRE2022- 102070, funded by MCIN/AEI/10.13039/501100011033 and by FSE+.

Declaration of competing interest

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

Data availability

All data and source code are freely available and open-sourced.

References

[1] H.T. Nguyen, P. Krishnan, D. Krishnaswamy, M. Usman, R. Buyya, Quantum cloud computing: A review, open problems, and future directions, 2024, http://dx.doi.org/10.48550/arXiv.2404.11420, arXiv preprint arXiv:2404.11420.

[2] T. Ichikawa, H. Hakoshima, K. Inui, K. Ito, R. Matsuda, K. Mitarai, et al., A comprehensive survey on quantum computer usage: How many qubits are employed for what purposes?, 2023, http://dx.doi.org/10.48550/arXiv.2307.16130, ArXiv:2307.16130.

[3] L. Liu, X. Dou, Qucloud+: A holistic qubit mapping scheme for single/multi-programming on 2D/3D NISQ quantum computers, ACM Trans. Archit. Code Optim. 21 (1) (2024) 1–27, http://dx.doi.org/10.1145/3631525.

[4] M.A. Serrano, J.A. Cruz-Lemus, R. Perez-Castillo, M. Piattini, Quantum software components and platforms: Overview and quality assessment, 55, (8) (ISSN: 0360-0300) 2022, http://dx.doi.org/10.1145/3548679.

[5] G. Bisicchia, J. García-Alonso, J.M. Murillo, A. Brogi, From quantum software handcrafting to quantum software engineering, in: 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering-Companion (SANER-C), 2024, pp. 149–150, http://dx.doi.org/10.1109/SANER-C62648.2024.00026.

[6] S. Nishio, R. Wakizaka, D. Sakuma, Y. Ueno, Y. Suzuki, Online job scheduler for fault-tolerant quantum multiprogramming, 2025, http://dx.doi.org/10.48550/arXiv.2505.06741, arXiv preprint arXiv:2505.06741.

[7] J. Alvarado-Valiente, J. Romero-Álvarez, J. Casco-Seco, E. Moguel, J. Garcia-Alonso, J.M. Murillo, Circuit scheduling policies on current QPUs: QCRAFT scheduler, in: International Conference on Service-Oriented Computing, Springer, 2024, pp. 195–209, http://dx.doi.org/10.1007/978-981-96-0808-9_15.

[8] J. Romero-Álvarez, J. Alvarado-Valiente, J. Casco-Seco, E. Moguel, J. Garcia-Alonso, J.M. Murillo, A noise validation for quantum circuit scheduling through a service-oriented architecture, Int. J. Softw. Eng. Knowl. Eng. 34 (09) (2024) 1371–1386, http://dx.doi.org/10.1142/S0218194024410018.

[9] G. Bisicchia, J. García-Alonso, J.M. Murillo, A. Brogi, Distributing quantum computations, by shots, in: International Conference on Service-Oriented Computing, 2023, pp. 363–377, http://dx.doi.org/10.1007/978-3-031-48421-6_25.

[10] G. Bisicchia, G. Clemente, J. Garcia-Alonso, J.M.M. Rodríguez, M. D'Elia, A. Brogi, Distributing quantum computations, shot-wise, 2024, http://dx.doi.org/10.48550/arXiv.2411.16530, arXiv preprint arXiv:2411.16530.

[11] M. Rahaman, M.M. Islam, An overview on quantum computing as a service (qcaas): Probability or possibility, Int. J. Math. Sci. Comput. (IJMSC) 2 (1) (2016) 16–22, http://dx.doi.org/10.5815/ijmsc.2016.01.02.

[12] V. Hassija, V. Chamola, V. Saxena, V. Chanana, P. Parashari, S. Mumtaz, M. Guizani, Present landscape of quantum computing, IET Quantum Commun. 1 (2) (2020) 42–48, http://dx.doi.org/10.1049/iet-qtc.2020.0027.

[13] J. Alvarado-Valiente, J. Romero-Álvarez, E. Moguel, J. García-Alonso, J.M. Murillo, Technological diversity of quantum computing providers: a comparative study and a proposal for API gateway integration, Softw. Qual. J. (2023) 1–21, http://dx.doi.org/10.1007/s11219-023-09633-5.

[14] C.Q. Choi, Ibm's quantum leap: The company will take quantum tech past the 1,000-qubit mark in 2023, IEEE Spectr. 60 (1) (2023) 46–47, http://dx.doi.org/10.1109/MSPEC.2023.10006669.

[15] H.T. Nguyen, M. Usman, R. Buyya, Qfaas: A serverless function-as-a-service framework for quantum computing, Future Gener. Comput. Syst. 154 (2024) 281–300, http://dx.doi.org/10.1016/j.future.2024.01.018.

[16] R.P. Garg, I.A. Sharapov, I. Sharapov, Techniques for optimizing applications: high performance computing, Prentice Hall PTR, USA, 2001, URL https://dl.acm.org/doi/10.5555/558986.

[17] F. Leymann, J. Barzen, The bitter truth about gate-based quantum algorithms in the NISQ era, Quantum Sci. Technol. 5 (4) (2020) 044007, http://dx.doi.org/10.1088/2058-9565/abae7d.

[18] S. Johnstun, J.-F. Van Huele, Understanding and compensating for noise on IBM quantum computers, Am. J. Phys. 89 (10) (2021) 935–942, http://dx.doi.org/10.1119/10.0006204.

[19] G. Bisicchia, J. García-Alonso, J.M. Murillo, A. Brogi, Dispatching shots among multiple quantum computers: An architectural proposal, in: 2023 IEEE International Conference on Quantum Computing and Engineering, QCE, 2, 2023, pp. 195–198, http://dx.doi.org/10.1109/QCE57702.2023.10210.

[20] G. Bisicchia, A. Bocci, J. García-Alonso, J.M. Murillo, A. Brogi, Cut&shoot: Cutting & distributing quantum circuits across multiple NISQ computers, in: 2024 IEEE International Conference on Quantum Computing and Engineering, QCE, 2, 2024, pp. 187–192, http://dx.doi.org/10.1109/QCE60285.2024.10276.

[21] T. Peng, et al., Simulating large quantum circuits on a small quantum computer, Phys. Rev. Lett. 125 (2020) 150504, http://dx.doi.org/10.1103/PhysRevLett.125.150504.

[22] W. Tang, et al., Cutqc: using small quantum computers for large quantum circuit evaluations, in: ASPLOS, 2021, pp. 473–486, http://dx.doi.org/10.1145/3445814.344675.

[23] G. Bisicchia, A. Bocci, A. Brogi, Quantum executor: A unified interface for quantum computing, 2025, http://dx.doi.org/10.48550/arXiv.2507.07597, arXiv preprint arXiv:2507.07597.

[24] N. Quetschlich, L. Burgholzer, R. Wille, MQT bench: Benchmarking software and design automation tools for quantum computing, Quantum 7 (2023) 1062, http://dx.doi.org/10.22331/q-2023-07-20-1062.

[25] Y. Cai, L.-H. Lim, Distances between probability distributions of different dimensions, IEEE Trans. Inform. Theory 68 (6) (2022) 4020–4031, http://dx.doi.org/10.1109/TIT.2022.3148923.

[26] S.I. Sazzad, P. Nag, Quantum noise and measuring quantum distance for NISQ circuits, Authorea Prepr. (2023) http://dx.doi.org/10.36227/techrxiv.21791987.v1.

[27] Y. Ohkura, T. Satoh, R. Van Meter, Simultaneous execution of quantum circuits on current and near-future NISQ systems, IEEE Trans. Quantum Eng. 3 (2022) 1–10, http://dx.doi.org/10.1109/TQE.2022.3164716.

[28] J. Alvarado-Valiente, J. Romero-Álvarez, J. Casco-Seco, E. Moguel, J. Garcia-Alonso, Quantum island mapping: Optimizing multi-circuit execution in quantum processors, in: Proceedings of the 2025 IEEE International Conference on Quantum Computing and Engineering, QCE, IEEE, 2025, pp. 791–796, http://dx.doi.org/10.1109/QCE65121.2025.00091.

[29] E. Moguel, J. Rojo, D. Valencia, J. Berrocal, J. Garcia-Alonso, J.M. Murillo, Quantum service-oriented computing: current landscape and challenges, Softw. Qual. J. 30 (4) (2022) 983–1002, http://dx.doi.org/10.1007/s11219-022-09589-y.

[30] J. Zhao, Quantum software engineering: Landscapes and horizons, 2020, http://dx.doi.org/10.48550/arXiv.2007.07047, arXiv preprint arXiv:2007.07047.

[31] J.M. Murillo, J. Garcia-Alonso, E. Moguel, J. Barzen, F. Leymann, S. Ali, T. Yue, P. Arcaini, R. Pérez-Castillo, I. García Rodríguez de Guzmán, M. Piattini, A. Ruiz-Cortes, A. Brogi, J. Zhao, A. Miranskyy, M. Wimmer, Quantum software engineering: Roadmap and challenges ahead, ACM Trans. Softw. Eng. Methodol. (2025) http://dx.doi.org/10.1145/3712002.

[32] A. McCaskey, E. Dumitrescu, D. Liakh, T. Humble, Hybrid programming for near-term quantum computing systems, in: 2018 IEEE International Conference on Rebooting Computing, ICRC, IEEE, 2018, pp. 1–12, http://dx.doi.org/10.1109/ICRC.2018.8638598.

[33] G.S. Ravi, K.N. Smith, P. Gokhale, F.T. Chong, Quantum computing in the cloud: Analyzing job and machine characteristics, in: 2021 IEEE International Symposium on Workload Characterization, IISWC, IEEE, 2021, pp. 39–50, http://dx.doi.org/10.48550/arXiv.2203.13121.

[34] H.T. Nguyen, M. Usman, R. Buyya, iQuantum: A case for modeling and simulation of quantum computing environments, in: 2023 IEEE International Conference on Quantum Software, IEEE Computer Society, Los Alamitos, CA, USA, 2023, pp. 21–30, http://dx.doi.org/10.1109/QSW59989.2023.00013.

[35] G.W. Dueck, A. Pathak, M.M. Rahman, A. Shukla, A. Banerjee, Optimization of circuits for ibm's five-qubit quantum computers, in: 2018 21st Euromicro Conference on Digital System Design, DSD, IEEE, 2018, pp. 680–684, http://dx.doi.org/10.1109/DSD.2018.00005.

[36] A. Kole, S. Hillmich, K. Datta, R. Wille, I. Sengupta, Improved mapping of quantum circuits to IBM QX architectures, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 39 (10) (2019) 2375–2383, http://dx.doi.org/10.1109/TCAD.2019.2962753.

[37] T. Fösel, M.Y. Niu, F. Marquardt, L. Li, Quantum circuit optimization with deep reinforcement learning, 2021, http://dx.doi.org/10.48550/arXiv.2103.07585, arXiv preprint arXiv:2103.07585.

[38] P. Das, S.S. Tannu, P.J. Nair, M. Qureshi, A case for multi-programming quantum computers, in: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, 2019, pp. 291–303, http://dx.doi.org/10.1145/3352460.3358287.

[39] A. Orenstein, V. Chaudhary, QGroup: Parallel quantum job scheduling using dynamic programming, in: 2024 IEEE International Conference on Quantum Computing and Engineering, QCE, 1, IEEE, 2024, pp. 990–999, http://dx.doi.org/10.1109/QCE60285.2024.00118.

[40] P. Seitz, M. Geiger, C. Ufrecht, A. Plinge, C. Mutschler, D. Scherer, C.B. Mendl, et al., SCIM MILQ: An HPC quantum scheduler, 2024, http://dx.doi.org/10.48550/arXiv.2404.03512, arXiv preprint arXiv:2404.03512.

[41] Y. Ohkura, T. Satoh, et al., Simultaneous execution of quantum circuits on current and near-future NISQ systems, IEEE Trans. Quantum Eng. 3 (2022) 1–10, http://dx.doi.org/10.1109/TQE.2022.3164716.

[42] J. Garcia-Alonso, J. Rojo, D. Valencia, E. Moguel, J. Berrocal, J.M. Murillo, Quantum software as a service through a quantum API gateway, IEEE Internet Comput. 26 (1) (2021) 34–41, http://dx.doi.org/10.1109/MIC.2021.3132688.

[43] M. Salm, J. Barzen, U. Breitenbücher, F. Leymann, B. Weder, K. Wild, The NISQ analyzer: automating the selection of quantum computers for quantum algorithms, in: Symposium and Summer School on Service-Oriented Computing, Springer, 2020, pp. 66–85, http://dx.doi.org/10.1007/978-3-030-64846-6_5.

[44] M. Salm, et al., Automating the comparison of quantum compilers for quantum circuits, in: CCIS, vol. 1429, 2021, pp. 64–80, http://dx.doi.org/10.1007/978-3-030-87568-8_4.

[45] M. Salm, et al., How to select quantum compilers and quantum computers before compilation, in: CLOSER, 2023, pp. 172–183, http://dx.doi.org/10.5220/0011775300003488.

[46] M. Salm, et al., Optimizing the prioritization of compiled quantum circuits by machine learning approaches, in: CCIS, vol. 1603, 2022, pp. 161–181, http://dx.doi.org/10.1007/978-3-031-18304-1_9.

[47] G.S. Ravi, et al., Adaptive job and resource management for the growing quantum cloud, in: IEEE QCE, 2021, pp. 301–312, http://dx.doi.org/10.1109/QCE52317.2021.00047.

[48] M. Grossi, et al., A serverless cloud integration for quantum computing, 2021, arXiv:2107.02007.

[49] N. Quetschlich, et al., Predicting good quantum circuit compilation options, 2022, CoRR, arXiv:2210.08027.

[50] S. Stein, N. Wiebe, Y. Ding, P. Bo, K. Kowalski, N. Baker, J. Ang, A. Li, Eqc: ensembled quantum computing for variational quantum algorithms, in: Proceedings of the 49th Annual International Symposium on Computer Architecture, 2022, pp. 59–71, http://dx.doi.org/10.1145/3470496.3527434.

[51] J. Li, Y. Song, Y. Liu, J. Pan, L. Yang, T. Humble, W. Jiang, QuSplit: Achieving both high fidelity and throughput via job splitting on noisy quantum computers, 2025, http://dx.doi.org/10.48550/arXiv.2501.12492, arXiv preprint arXiv:2501.12492.