Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

CQ-Migrate: A Hardware-Aware Feasibility Framework for NISQ Workloads with Multi-Run Validation onTuna-17

Ravikumar Sreepada

sravikumar_ece@mvsrec.edu.in

MVSR Engineering College

Mahendar Gajula

MVSR Engineering College

Sarath Chandra Maturi

MVSR Engineering College

Research Article

Keywords: quantum computing, NISQ, Grover search, workload migration, hybrid quantum--classical computing, Quantum Inspire, benchmarking, reproducibility

Posted Date: November 9th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11163491/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

CQ-Migrate: A Hardware-Aware Feasibility Framework for NISQ Workloads with Multi-Run Validation onTuna-17

Ravikumar Sreepada[1,2*], Mahendar Gajula[2,3†] and Sarath Chandra Maturi[1,2†]

[1*]Department of ECE, MVSR Engineering College, Nadergul, Hyderabad, 501510, Telangana, India.
[2]Department of ECE, MVSR Engineering College, Nadergul, Hyderabad, 501510, Telangana, India.
[3]Department of ECE, MVSR Engineering College, Nadergul, Hyderabad, 501510, Telangana, India.

*Corresponding author(s). E-mail(s): sravikumar_ece@mvsrec.edu.in;
Contributing authors: mahendar_ece@mvsrec.edu.in;
sarathchandra_ece@mvsrec.edu.in;
†These authors contributed equally to this work.

Abstract

Classical-to-quantum workload migration is not a direct code-translation task because theoretical speedup, reversible mapping, circuit resources, device noise, compilation, queueing, and classical–quantum orchestration must be considered together. This work presents CQ-Migrate, a structured decision framework for determining whether a workload should remain classical, be decomposed into a hybrid workflow, or be evaluated on a quantum processing unit. Grover search is used as an empirical case study. A true two-qubit Grover circuit was validated locally, executed on the Quantum Inspire QX emulator, and then evaluated on the 17-qubit Tuna-17 superconducting processor. Four marked states were tested with five independent hardware runs per state and 1024 shots per run, totaling 20,480 hardware shots. The emulator returned the marked state in 100% of 4096 shots. Tuna-17 produced 19,076 correct measurements, corresponding to an aggregate success probability of 93.14% (95% Wilson interval: 92.79–93.48%). State-wise mean success ranged from 91.82% to 94.30%; run-level differences across marked states were statistically detectable (one-way ANOVA $\boldsymbol{F = 5.99}$, $\boldsymbol{p = 0.0062}$; Kruskal–Wallis $\boldsymbol{H = 10.88}$, $\boldsymbol{p = 0.0124}$). The study also reports

complete provider job provenance and reinterprets earlier archived IBM executions as a backend-variability control rather than a Grover validation. The results support evidence-driven, hardware-aware migration decisions without claiming practical quantum speedup for the small instance studied.

Keywords: quantum computing; NISQ; Grover search; workload migration; hybrid quantum–classical computing; Quantum Inspire; benchmarking; reproducibility

1 Introduction

Quantum algorithms can offer substantial asymptotic improvements for selected computational problems, yet theoretical complexity alone does not determine whether a real workload should be moved to a quantum processor. Current quantum processing units (QPUs) remain subject to noise, finite coherence, imperfect measurements, topology constraints, compilation overhead, service queues, and the cost of coordinating classical and quantum components. Preskill's NISQ perspective therefore remains relevant: near-term systems are valuable experimental platforms, but useful computation requires careful matching of algorithms, circuits, and hardware constraints [1].

This mismatch between theoretical speedup and deployable benefit creates a systems-engineering question: *when should a classical workload be retained, decomposed into a hybrid workflow, or mapped to a QPU?* Quantum software engineering studies now identify architecture, interoperability, testing, observability, and reproducibility as first-class concerns [2]. Benchmarking research similarly emphasizes that a useful performance evaluation must account for more than a single device metric [3, 4].

This paper addresses that decision problem through *CQ-Migrate*, a five-stage framework that begins with a classical workload and evaluates quantum suitability, reversible transformability, hardware feasibility, hybrid decomposition, and empirical evidence. Grover's unstructured search algorithm [5] is used as the case study because it provides a clear theoretical reduction in oracle-query complexity while remaining highly sensitive to implementation depth and hardware errors. Recent studies have used Grover search both to study real-hardware performance and to benchmark quantum devices [6–8].

The present work extends an earlier simulator-centered manuscript in two important ways. First, a true two-qubit Grover experiment is executed on real superconducting hardware using Quantum Inspire's Tuna-17 processor. Second, all four marked states are evaluated repeatedly, with provider job identifiers, result identifiers, raw counts, compilation depth, and client-observed wall time retained for each run. A QX emulator reference provides the noise-free platform baseline. Earlier archived IBM executions from the project are retained only as a secondary backend-variability control because the preserved circuit is not a complete Grover iterate.

The contributions of this paper are:

1. a structured CQ-Migrate workflow that treats classical-to-quantum migration as a multi-criteria engineering decision rather than source-to-source translation;
2. a reproducible two-qubit Grover implementation and experimental protocol for Quantum Inspire hardware;
3. a repeated real-hardware study comprising 20 Tuna-17 executions and 20,480 hardware shots across all four possible marked states;
4. statistical characterization of success probability, run-to-run variation, state-dependent behavior, and error structure, with a noise-free QX emulator reference;
5. complete experiment provenance through Quantum Inspire job, batch-job, and result identifiers, together with checkpointed scripts and raw CSV data; and
6. an evidence correction of earlier IBM results, separating genuine Grover validation from a shallow cross-backend control circuit.

The work does not claim practical quantum speedup for the $N = 4$ hardware instance. Rather, it evaluates the reliability and reproducibility of the migrated circuit and uses the observations to inform a broader migration framework.

2 Background and Related Work

2.1 Grover Search and Amplitude Amplification

For an unstructured search space of size $N = 2^n$ containing one marked item, a classical linear search requires $O(N)$ queries in the worst case and $(N+1)/2$ comparisons on average when the marked position is uniformly distributed. Grover's algorithm reduces the oracle-query complexity to $O(\sqrt{N})$ [5]. The initial uniform superposition is

$$|s\rangle = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} |x\rangle. \tag{1}$$

A phase oracle O_f changes the phase of the marked state $|w\rangle$,

$$O_f|x\rangle = (-1)^{f(x)}|x\rangle, \tag{2}$$

where $f(w) = 1$ and $f(x) = 0$ otherwise. The diffusion operator is

$$D = 2|s\rangle\langle s| - I, \tag{3}$$

and one Grover iteration is $G = DO_f$. If $\theta = \sin^{-1}(1/\sqrt{N})$, then after r iterations the ideal success probability is

$$P_{\text{ideal}}(r) = \sin^2\left((2r+1)\theta\right). \tag{4}$$

For $N = 4$, a single iteration is optimal and ideally produces the marked state with probability one.

2.2 Grover Search on NISQ Hardware

The quadratic query advantage does not guarantee a wall-clock advantage. Real hardware must implement reversible oracle logic, basis transformations, two-qubit interactions, routing, and measurement. AbuGhanem characterized Grover search on superconducting devices and demonstrated the importance of hardware-aware fidelity analysis [6]. Pokharel and Lidar showed that error detection and suppression can enable better-than-classical Grover behavior for small instances [7]. GRADE uses generalized Grover circuits as a reliability benchmark across hardware platforms [8]. Rosmanis analyzed the extent to which classical computation can assist quantum search and reinforced the fundamental role of the quantum component in achieving Grover-type speedups [9]. Distributed variants further show how hardware limits can motivate algorithmic decomposition [10].

2.3 Hybrid Systems, Benchmarking, and Quantum Software Engineering

Classical-to-quantum migration has characteristics of both algorithm design and systems integration. Yao and Gupta proposed benchmarking of hybrid quantum–classical edge-cloud environments using communication, transpilation, and execution latency [4]. Acuaviva *et al.* argued for standardized quantum-computer performance evaluation and emphasized benchmark quality attributes beyond isolated metrics [3]. Quantum software engineering research identifies hybrid architecture, debugging, testing, interoperability, maintenance, and reproducibility as persistent challenges [2]. These studies motivate an explicit migration workflow in which algorithmic suitability and systems feasibility are evaluated together.

3 CQ-Migrate Framework

CQ-Migrate treats migration as an iterative mapping from a classical workload W_C to an implementable quantum or hybrid workflow W_Q. Figure 1 summarizes the process.

3.1 Stage 1: Workload Classification

The workload is classified by computational structure (e.g., search, optimization, simulation, linear algebra, cryptography, machine learning, or control). A quantum mapping is considered only if an algorithmic primitive with a credible theoretical advantage is known.

3.2 Stage 2: Reversible Mapping

Classical predicates and transformations must be expressed as reversible unitary operations. For search, the key engineering object is the oracle. The cost of building the

Table 1 Positioning of CQ-Migrate relative to representative recent work

Study	Primary focus	Hardware/system factors	Difference from CQ-Migrate
AbuGhanem [6]	Grover characterization	Superconducting hardware, fidelity	Algorithm/device characterization after workload choice
Pokharel and Lidar [7]	Grover with error suppression	Real hardware, mitigation	Search performance rather than general workload-placement decision
GRADE [8]	Grover reliability benchmark	Cross-platform accuracy/stability	Benchmarks hardware after an algorithm has been selected
Yao and Gupta [4]	Hybrid edge-cloud benchmark	Communication, transpilation, execution latency	Strong systems benchmark; less emphasis on reversible transformability of an arbitrary classical workload
Murillo *et al.* [2]	Quantum software engineering roadmap	Architecture, testing, lifecycle	Broad engineering agenda rather than an empirical migration decision workflow
CQ-Migrate	Classical-to-quantum placement	Transformability, resources, hardware success, orchestration, provenance	Begins with the classical workload and selects classical, hybrid, or quantum placement

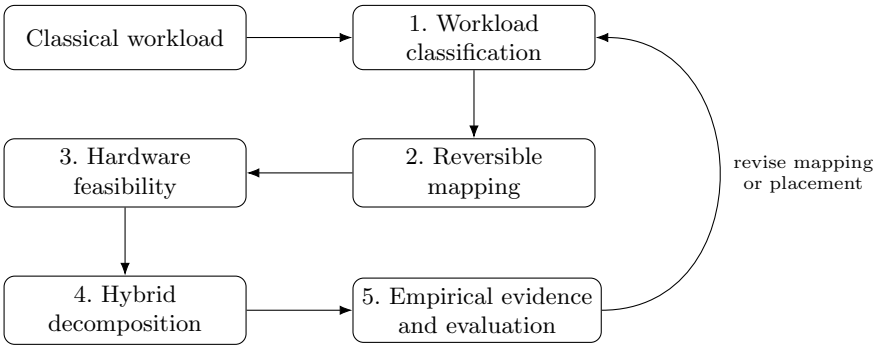


Fig. 1 CQ-Migrate workflow. Empirical results can return the workload to an earlier stage rather than forcing quantum execution.

oracle is part of the migration decision because an expensive reversible implementation can dominate the nominal query advantage.

3.3 Stage 3: Hardware Feasibility

The mapped circuit is evaluated against backend constraints: required qubits, topology, supported gates, compiled depth, two-qubit operations, measurement behavior,

and expected fidelity. The qubit adequacy condition is

$$Q_{\text{req}} \leq Q_{\text{avail}}. \tag{5}$$

The algorithmic query gain may be represented as

$$G_Q(N) = \frac{Q_C(N)}{Q_Q(N)}, \tag{6}$$

where Q_C and Q_Q are classical and quantum query counts. $G_Q > 1$ is necessary for an algorithmic advantage but is not sufficient for practical migration.

3.4 Stage 4: Hybrid Decomposition

Near-term workflows are generally hybrid. A useful end-to-end decomposition is

$$T_{\text{obs}} = T_{\text{encode}} + T_{\text{transpile}} + T_{\text{queue}} + T_{\text{QPU}} + T_{\text{measure}} + T_{\text{return}} + T_{\text{post}}. \tag{7}$$

A client can often observe only the sum of these terms. For this reason, client wall time must not be mislabeled as QPU execution time.

3.5 Stage 5: Empirical Evidence and Decision

The final placement decision uses empirical reliability and service requirements rather than a universal scalar threshold. The user defines an acceptable success threshold τ_P and, where relevant, an end-to-end latency threshold τ_T. Algorithm 1 summarizes the decision procedure.

4 Grover Case Study and Experimental Algorithm

4.1 Two-Qubit Oracle Construction

For the four-state search space $\{00, 01, 10, 11\}$, qubits q_0 and q_1 are initialized with Hadamard gates. The phase oracle is realized with a controlled-Z operation. If a bit of the marked state is zero, an X gate is applied to the corresponding qubit before the controlled-Z and undone afterward. The diffusion operator is implemented as

$$H^{\otimes 2} X^{\otimes 2} CZ X^{\otimes 2} H^{\otimes 2}. \tag{8}$$

Only one Grover iteration is required for $N = 4$.

4.2 Hardware Experiment Algorithm

Algorithm 2 gives the procedure used for both the QX emulator and Tuna-17. The hardware backend requires the program to declare the full processor width even if only a subset of qubits is actively used [11]. Therefore, the Tuna-17 circuit declares 17 qubits while the Grover computation and measurements use two logical qubits.

Algorithm 1 CQ-Migrate decision procedure

Require: Classical workload W, candidate backend B, reliability threshold τ_P, optional latency threshold τ_T

Classify W and identify a quantum primitive with a defensible theoretical gain
if no suitable quantum primitive exists **then**
    return CLASSICAL
end if
Construct or estimate the reversible mapping and oracle cost
if mapping/resource requirements exceed backend constraints **then**
    return CLASSICAL or HYBRID-DECOMPOSE
end if
Transpile for B and record depth, size, and relevant two-qubit operations
Execute simulator/emulator reference and repeated hardware trials
Estimate $\hat{P}$, uncertainty, run-to-run variation, and T_{obs}
if $\hat{P} < \tau_P$ **then**
    return CLASSICAL or HYBRID-DECOMPOSE
else if τ_T is defined and $T_{\text{obs}} > \tau_T$ **then**
    return HYBRID
else
    return QUANTUM-CANDIDATE, with empirical evidence attached
end if

Quantum Inspire compiles submitted programs for the selected hardware topology and gate set [11, 12].

5 Experimental Methodology

5.1 Software and Access Workflow

Experiments were conducted from a Windows workstation using a Python 3.12 virtual environment. Qiskit was used to construct and transpile circuits [13], and the Qiskit-QuantumInspire provider was used to enumerate backends and submit circuits programmatically [14]. Authentication was completed with the Quantum Inspire command-line login workflow. Provider access was verified by listing available backends before any hardware jobs were submitted.

The accessible list at the time of the study included QX emulator, Ry emulator, Tuna-5, Tuna-9, and Tuna-17; Tuna-17 was the available real superconducting backend used for the final study. Quantum Inspire describes Tuna-17 as a 17-transmon processor arranged in a "ninja-star" topology with 24 flux-tunable nearest-neighbor couplers, available since April 2026 [11]. Its compiler decomposes submitted operations to the device's supported gate set and schedules the resulting operations [12].

5.2 Experiment Process

Figure 2 documents the process actually followed in this work.

Algorithm 2 Repeated two-qubit Grover experiment on Quantum Inspire

Require: Backend B, marked state $m \in \{00, 01, 10, 11\}$, shots S, repeats R

Set circuit width to Q_B, the full number of qubits on backend B
Allocate Q_B quantum bits and two classical measurement bits
Apply H to q_0, q_1
for each target bit equal to 0 **do**
    Apply X to the corresponding qubit
end for
Apply $CZ(q_0, q_1)$ and undo the target-conditioning X gates
Apply diffusion $H^{\otimes 2} X^{\otimes 2} CZ X^{\otimes 2} H^{\otimes 2}$
Insert a barrier and measure q_0, q_1
Transpile for B with optimization level 1 and fixed transpiler seed 42
for $r = 1$ to R **do**
    Submit S shots and wait for the provider result
    Store raw counts, Quantum Inspire job/batch/result IDs, timestamps, compiled depth/size, and client wall time
    Compute $\hat{P}_m = C_m/S$ and a 95% Wilson interval
    Checkpoint the run immediately to CSV and raw JSON/QPY evidence
end for
return Complete provenance and statistical record

5.3 Local Validation

Before cloud execution, the ideal statevector behavior of the two-qubit circuit was checked for all four possible marked states. Each target produced ideal probability numerically equal to one for the marked basis state, with residual probabilities on other states at floating-point roundoff levels. This step prevented hardware time from being used on an incorrectly specified oracle or diffusion operator.

5.4 QX Emulator Reference

The noise-free reference used Quantum Inspire's QX emulator through the same provider interface. Each marked state was executed once with 1024 shots. The four emulator jobs returned the marked state in every shot. The provider job IDs were 1443235–1443238 and result IDs were 1406896–1406899. The same state-specific compiled depths and sizes were observed as in the hardware study.

5.5 Tuna-17 Pilot and Full Hardware Study

A pilot run targeted $|11\rangle$ with 1024 shots to validate the complete submission, execution, and evidence-capture chain. The provider dashboard reported Job 1441927 and Result 1405596, with 965 marked-state counts (94.24% success); this pilot was *excluded* from the confirmatory 20-run dataset.

The final hardware experiment used all four marked states. Each state was executed in five independent jobs with 1024 shots, producing 20 jobs and 20,480 hardware shots. Runs were recorded between 12 September 2026 16:38 UTC and 13 September 2026

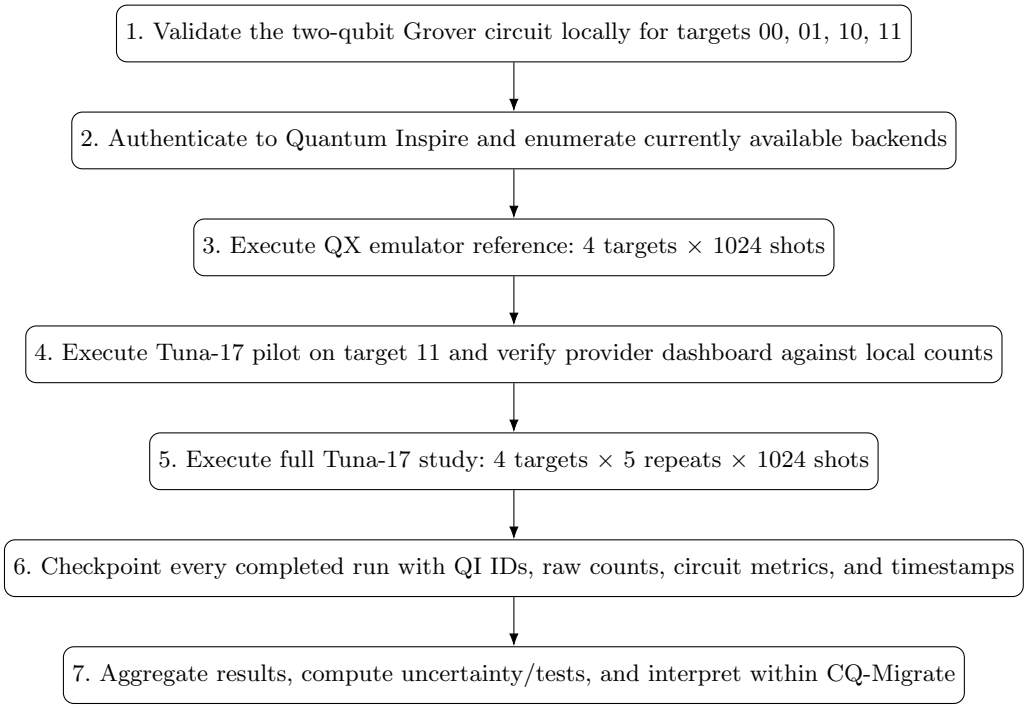


Fig. 2 Experimental workflow used in the study. A resume-safe script was used after a transient network interruption so completed provider jobs were not duplicated.

03:53 UTC. The full experiment generated unique Quantum Inspire job, batch-job, and result identifiers for every run. The complete identifiers are supplied in Supplementary Data rather than compressed into the main manuscript.

5.6 Statistical Analysis

For each run, success probability was defined as

$$\hat{P}_m = \frac{C_m}{S}, \tag{9}$$

where C_m is the count of the marked state and $S = 1024$. Wilson 95% binomial intervals were computed for pooled shot-level proportions. Because the five runs per target can reflect time-dependent hardware variation, run-level success probabilities were also summarized by mean and standard deviation. A one-way ANOVA tested differences among the four target-state run means; the Kruskal–Wallis test was used as a non-parametric sensitivity check. Tukey's honestly significant difference test was used only as an exploratory post-hoc comparison. The statistical unit for these between-state tests was the run, not the individual shot.

Table 2 Final experimental configuration

Parameter	Setting
Problem	One marked item in $N = 4$ unstructured search
Marked states	00, 01, 10, 11
Grover iterations	1
Reference backend	Quantum Inspire QX emulator (10 qubits exposed)
Hardware backend	Quantum Inspire Tuna-17
Hardware technology	17 superconducting transmon qubits
Hardware repeats	5 independent jobs per marked state
Shots per job	1024
Total hardware jobs	20
Total hardware shots	20,480
Emulator jobs/shots	4 jobs / 4096 shots
Transpilation	Qiskit, optimization level 1, seed 42
Recorded provenance	job, batch-job, result IDs; UTC times; counts; depth; size; wall time

6 Results

6.1 Theoretical Query Scaling

Figure 3 shows the expected query growth. The classical expected comparisons increase linearly, whereas Grover iterations increase approximately as $\sqrt{N}$. For the $N = 4$ experiment itself, the classical expected number of comparisons is 2.5 and one Grover oracle iteration is sufficient. This query advantage is algorithmic; no wall-clock quantum speedup is claimed for the small cloud-hardware experiment.

6.2 QX Emulator Reference

The QX emulator returned the marked state in 1024 of 1024 shots for each of the four targets. Thus the aggregate emulator result was 4096 correct outcomes in 4096 shots. The Wilson interval for each 1024/1024 result is approximately 0.9963–1.0000. This result confirms the circuit logic and provides a platform reference against which real-hardware degradation can be interpreted.

6.3 Tuna-17 Hardware Success

Across all 20 hardware jobs, 19,076 of 20,480 shots returned the intended marked state. The aggregate success probability was

$$\hat{P}_{\text{Tuna17}} = 0.931445, \tag{10}$$

with pooled Wilson 95% interval 0.9279–0.9348. Individual run success ranged from 0.8994 to 0.9512.

Table 3 reports the state-wise results. Target 00 produced the highest mean success (94.30%), while target 01 produced the lowest (91.82%) and the largest run-to-run

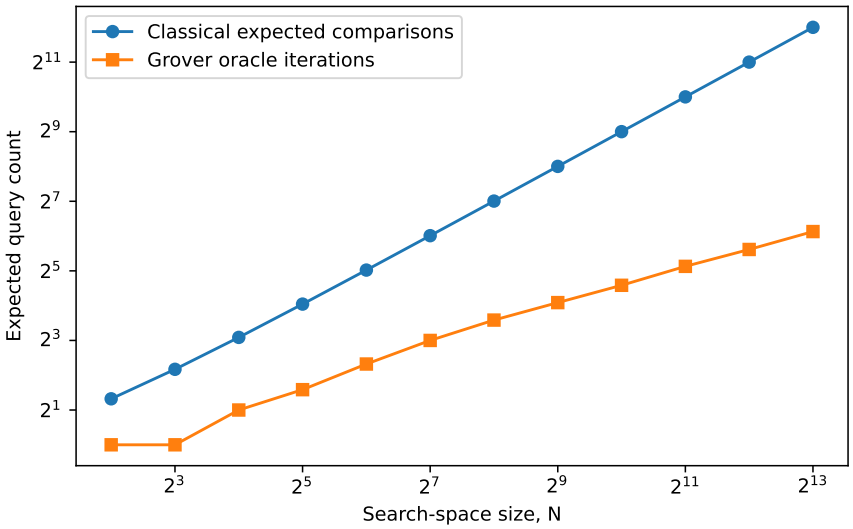


Fig. 3 Classical expected comparisons and Grover oracle iterations versus search-space size.

Table 3 Tuna-17 success by marked state (five runs, 5120 shots per state)

Target	Correct	Mean P	SD	Pooled 95% CI	Depth	Size
00	4828	0.9430	0.0049	[0.9363, 0.9490]	7	12
01	4701	0.9182	0.0163	[0.9103, 0.9254]	7	11
10	4769	0.9314	0.0064	[0.9242, 0.9380]	7	11
11	4778	0.9332	0.0042	[0.9260, 0.9397]	6	10

standard deviation (1.63 percentage points). The compiled circuit for target 11 was slightly smaller and shallower (depth 6, size 10) than the other target circuits.

Figure 4 shows the run-level means with 95% t intervals across the five repeats; the dashed line marks the emulator reference. Figure 5 shows all 20 run-level success values.

6.4 State-Dependent Variation

A one-way ANOVA over the 20 run-level probabilities found a difference among marked-state groups ($F = 5.9888$, $p = 0.00617$). The non-parametric Kruskal–Wallis test gave the same qualitative conclusion ($H = 10.8798$, $p = 0.01239$). An exploratory Tukey comparison found the clearest separation between targets 00 and 01 ($p \approx 0.003$); the remaining pairwise comparisons were not significant at 0.05 after Tukey adjustment.

These differences should not be interpreted as a permanent state ranking. The experiment did not capture per-job calibration snapshots, and the jobs were

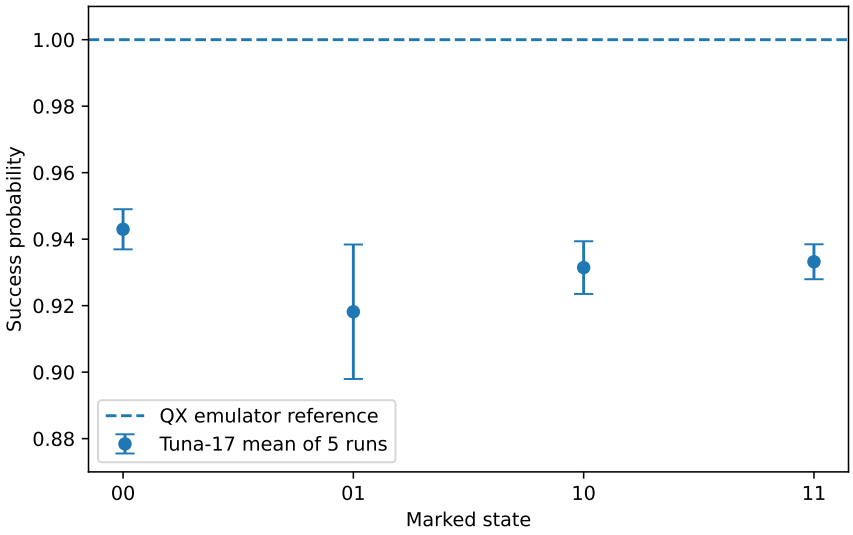


Fig. 4 Tuna-17 mean success across five runs per target, with 95% t intervals over run-level probabilities. The QX emulator returned 1.0 for every target.

Table 4 Aggregated Tuna-17 measurement counts across five runs per marked state

Marked state	00	01	10	11
00	4828	81	156	55
01	148	4701	84	187
10	209	49	4769	93
11	33	197	112	4778

distributed over several hours. The result instead shows that state-specific and time-dependent variability is large enough to be measurable even in a very small Grover circuit.

6.5 Error Structure

Table 4 and Fig. 6 show the aggregated hardware outcome distribution. Of the 1404 incorrect hardware shots, 1183 (5.78% of all shots) were one-bit Hamming-distance errors and 221 (1.08% of all shots) differed from the target in both bits. The dominance of one-bit errors is consistent with the presence of asymmetric control/readout error channels, but the present study does not contain sufficient calibration data to attribute the errors to a specific physical mechanism.

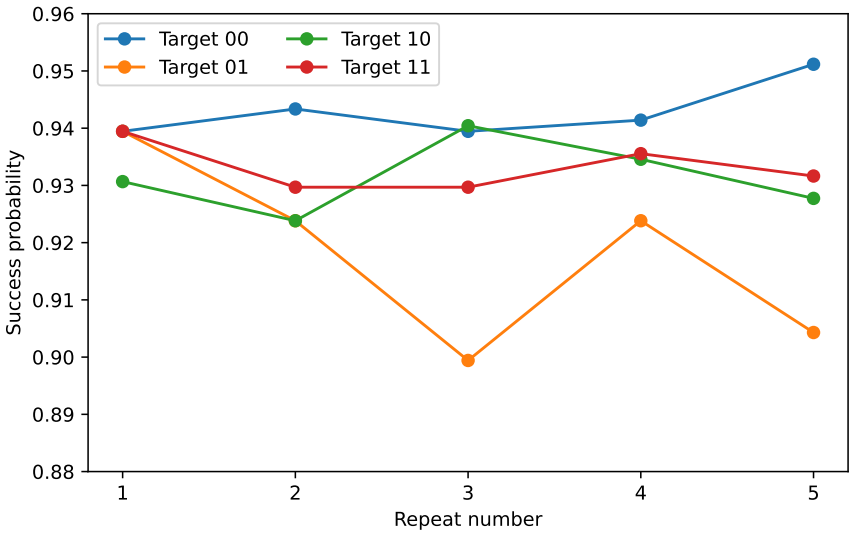


Fig. 5 Run-to-run Tuna-17 success probability for the four marked states.

Table 5 Archived IBM control-experiment counts (1024 shots per backend)

Backend	00	01	10	11	χ^2	p
`ibm_kingston`	260	260	269	235	2.508	0.474
`ibm_marrakesh`	243	267	276	238	3.961	0.266
`ibm_fez`	222	247	310	245	16.695	0.00082

6.6 Observed Cloud-Orchestration Time

Client-observed wall time varied widely across the hardware runs. Median wall times were approximately 205.5 s (target 00), 574.0 s (01), 687.9 s (10), and 88.6 s (11), with a maximum of 2107 s. These values include queue, network, provider service, compilation/dispatch, result retrieval, and local overhead and therefore cannot be interpreted as QPU execution time. The large spread reinforces the CQ-Migrate requirement to separate algorithmic gain from end-to-end service behavior. A run-level wall-time plot is supplied as Supplementary Fig. S1.

7 Archived IBM Backend-Variability Control

The earlier phase of this project preserved terminal evidence from a two-qubit phase/interference circuit executed on `ibm_fez`, `ibm_marrakesh`, and `ibm_kingston`. Inspection showed that the circuit was not a complete Grover search; therefore it is not included in the Grover hardware success claims. It remains useful as an archived provider-variability control. Exact terminal-preserved counts are shown in Table 5.

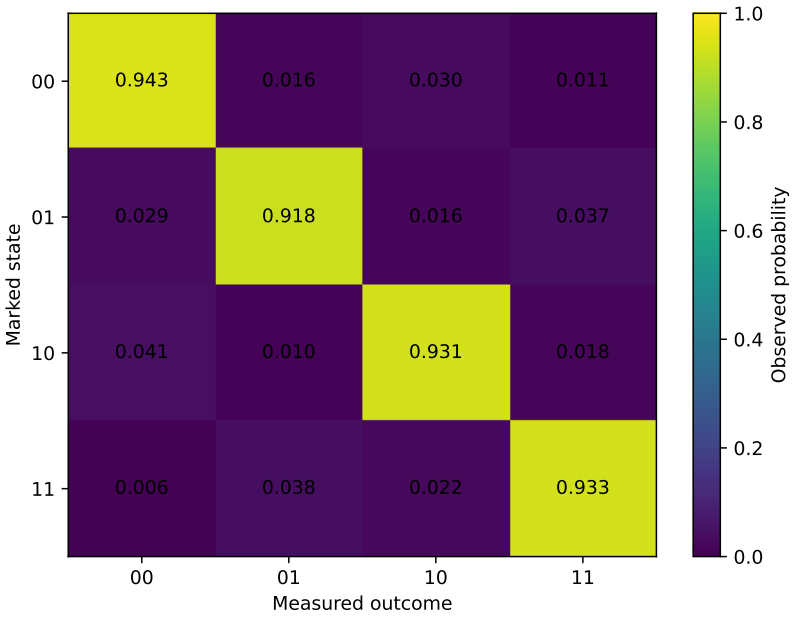


Fig. 6 Normalized measurement distribution on Tuna-17. Rows denote the marked state; columns denote the measured outcome.

Only the Fez distribution departed significantly from an ideal uniform four-state reference in this single archived run. Because there was only one execution per backend and no synchronized calibration record, the result cannot establish a stable provider ranking. The re-analysis is included to demonstrate evidence hygiene: a migration study should preserve the circuit definition and distinguish a generic hardware control from algorithm-specific validation.

8 Discussion

8.1 Implications for Classical-to-Quantum Migration

The experiments illustrate why migration should be evaluated in layers. First, theory identifies a candidate algorithmic advantage. Second, circuit synthesis determines whether the quantum primitive is implementable within resource limits. Third, real hardware determines whether the mapped computation remains reliable. Finally, the cloud service and classical control path determine whether the complete workflow satisfies application constraints.

For the present $N = 4$ case, the algorithmic component is exact on the QX emulator and remains highly reliable on Tuna-17, with approximately 93% aggregate hardware success. This is sufficient to establish *execution feasibility* and reproducibility of the migrated quantum subroutine. It does not establish economic or runtime superiority over a classical search, because a four-element classical search is trivial and cloud orchestration dominates end-to-end time.

8.2 Why Repetition and Provenance Matter

The five-run design revealed information that a single 1024-shot histogram would miss. The target-01 success rate varied from 89.94% to 93.95%, while target 00 was considerably more stable. Repeated independent jobs therefore expose drift and service-time variation in addition to within-job sampling error. Retaining the provider job and result identifiers makes the experiment auditable and permits comparison with the provider dashboard.

A transient network interruption occurred during data collection. Because every completed result had been checkpointed and the resume script skipped completed state/repeat combinations, the study could continue without discarding valid runs or silently duplicating them. This operational detail is relevant to reproducible cloud-QPU experimentation and is included in the released code.

8.3 Hardware-Aware Interpretation of State Dependence

The marked state changes the oracle conditioning and hence the exact compiled instruction sequence. In this study, target 11 compiled to depth 6 and size 10, whereas the other targets required depth 7 and size 11–12. The state groups also exhibited statistically detectable differences. These observations suggest that workload mapping and compiled structure should be treated as part of migration evidence, not as incidental implementation details. However, without synchronized calibration and qubit-level error data, causal claims about individual physical errors would be premature.

8.4 Relevance to CQ-Migrate

The final CQ-Migrate decision is intentionally multi-criteria. A favorable asymptotic gain can coexist with unacceptable hardware error or orchestration overhead. Conversely, high circuit fidelity on a tiny instance does not imply useful quantum advantage at scale. The correct engineering response may therefore be to keep the full workload classical, to move only a quantum-suitable kernel, or to defer migration until the target hardware can satisfy an application-defined reliability/latency threshold.

9 Threats to Validity

Problem size. The real-hardware case uses only two active qubits and $N = 4$. It demonstrates correctness and reliability of a migrated quantum subroutine, not practical speedup or scalability. Larger search spaces require deeper or more complex oracles and are expected to be more noise-sensitive.

Single real-hardware backend. The confirmatory Grover study uses Tuna-17 only. Tuna-5 and Tuna-9 were unavailable when the experiment was conducted. The archived IBM data provide provider diversity but not a matched Grover comparison. A future study should repeat the same Grover protocol on additional backends under contemporaneous conditions.

Calibration metadata. Per-job qubit calibration snapshots were not captured. State-dependent variation therefore cannot be uniquely attributed to readout, single-qubit, two-qubit, or drift mechanisms.

Timing metric. The recorded wall time is client-observed end-to-end time, not pure QPU duration. It is useful for cloud-service feasibility but should not be used to infer processor execution speed.

Statistical dependence. Shot-level Wilson intervals treat outcomes within a pooled state group as binomial observations, while hardware drift may create extra-binomial variation. The primary between-state tests therefore use run-level probabilities. Five runs per state remain a modest sample and post-hoc results should be considered exploratory.

Generality of CQ-Migrate. Grover search is only one workload family. External validation on optimization (e.g., MaxCut/QAOA), SAT/oracle synthesis, or quantum machine-learning kernels would strengthen the general framework.

10 Reproducibility and Data Availability

The submission package contains the two canonical result files (Tuna-17 hardware and QX emulator), a complete job-provenance table, the resume-safe Quantum Inspire execution script, the analysis script used to regenerate statistics and figures, and the original evidence screenshot for the pilot job. The Qiskit-QuantumInspire provider supports programmatic backend selection and circuit submission through `QIProvider` and `backend.run` [14]. Quantum Inspire also documents the hardware compilation requirements, including declaration of the full hardware qubit count and hardware-specific decomposition [12, 15]. No authentication credential is included in the artifacts.

The primary data supporting this article are supplied as Supplementary Data S1 (`qi_grover_tuna17_full.csv`) and Supplementary Data S2 (`qi_grover_qx_reference.csv`). Supplementary Data S3 contains job-level provenance. Supplementary Code S1 reproduces the hardware/emulator experiment, and Supplementary Code S2 reproduces the analysis and figures. A public repository DOI may be added at submission or revision if required by the editor.

11 Conclusion

This work developed and empirically exercised CQ-Migrate, a framework for evidence-driven classical-to-quantum workload migration. Grover search was used to connect theoretical query advantage to reversible mapping, hardware compilation, repeated real-QPU execution, uncertainty analysis, and cloud-service overhead. The QX emulator produced perfect marked-state recovery for all four two-qubit targets, while 20 Tuna-17 hardware jobs produced an aggregate success probability of 93.14% across 20,480 shots. The experiment also exposed statistically detectable state-dependent variation, a predominance of one-bit errors, and large variation in client-observed service time.

The main conclusion is methodological: a successful quantum migration claim should contain more than an asymptotic complexity argument or a single successful hardware job. It should preserve the exact circuit, compilation characteristics, repeated hardware results, uncertainty, provider provenance, and end-to-end systems context. Under CQ-Migrate, these measurements determine whether a workload is ready for

quantum placement, better suited to a hybrid decomposition, or should remain classical. Future work will extend the same protocol to larger Grover/SAT instances, additional hardware backends, and an optimization workload so that the framework can be validated beyond unstructured search.

Declarations

Funding

Not applicable. This research received no external funding.

Clinical Trial Registration

Not applicable. This study is not a clinical trial.

Consent to Publish

Not applicable.

Ethics Approval and Consent to Participate

Not applicable. This study did not involve human participants, human data, or animals.

References

[1] Preskill, J.: Quantum computing in the NISQ era and beyond. Quantum **2**, 79 (2018)

[2] Juan M. Murillo, J.G.-A., *et al.*: Quantum software engineering: Roadmap and challenges ahead. ACM Transactions on Software Engineering and Methodology, 2025 **34**, 154–115448 (2025) https://doi.org/10.1145/3712002

[3] Acuaviva, A., Aguirre, D., Pena, R., Sanz, M.: Benchmarking quantum computers:towards a standard performance evaluation approach. Quantum Sciance and Technology **11**(2), 1–38 (2026) https://doi.org/10.1088/2058-9565/ae4269

[4] Yao, G., Gupta, L.: A benchmarking framework for hybrid quantum-classical edge-cloud computing systems. Applied Sciences **15**(18), 1–35 (2025) https://doi.org/10.3390/app151810245

[5] Grover, L.K.: A Fast Quantum Mechanical Algorithm for Database Search. Paper presented at the 28th Annual ACM Symposium on the Theory of Computing (STOC), May 1996 (1996)

[6] AbuGhanem, M.: Characterizing grover search algorithm on large-scale superconducting quantum computers. Scientific Reports (2025) https://doi.org/10.1038/s41598-024-80188-6

[7] Pokharel, B., Lidar, D.: Better-than-classical grover search via quantum error detection and suppression. npj Quantum Information **10** (2024) https://doi.org/10.1038/s41534-023-00794-6

[8] Khalid Azain, M., Oliver, Z.: Grade: Grover-based benchmarking toolkit for assessing quantum hardware. arXiv:2504.19387 (2025) https://doi.org/10.48550/arXiv.2504.19387

[9] Rosmanis, A.: Hybrid quantum-classical search algorithms. ACM Transactions on Quantum Computing **5** (2024) https://doi.org/10.1145/3648573

[10] Daowen, Q., Luo, L., Ligang, X.: Distributed grover's algorithm. Theoretical Computer Science **993** (2024) https://doi.org/10.1016/j.tcs.2024.114461

[11] Quantum Inspire: Transmons: Tuna-17 backend. https://www.quantum-inspire.com/backends/transmons/. Accessed 13 Sep 2026 (2026)

[12] Quantum Inspire: Tuna backends: operational specifics. https://www.quantum-inspire.com/kbase/tuna-operational-specifics/. Accessed 13 Sep 2026 (2026)

[13] Quantum Inspire: Compiling and executing your algorithm. https://www.quantum-inspire.com/kbase/executing-your-algorithm/. Accessed 13 Sep 2026 (2026)

[14] QuTech: Qiskit-QuantumInspire documentation: submitting a circuit. https://qutech-delft.github.io/qiskit-quantuminspire/getting_started/submitting.html. Version 0.18.2 documentation; accessed 13 Sep 2026 (2026)

[15] QuTech: QuantumInspire. Transmons: Tuna17 backend. QuTech. https://qutech-delft.github.io/qiskit-quantuminspire/getting_started/submitting.html. accessed 13 Sep 2026 (2026)

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- supplementary.zip