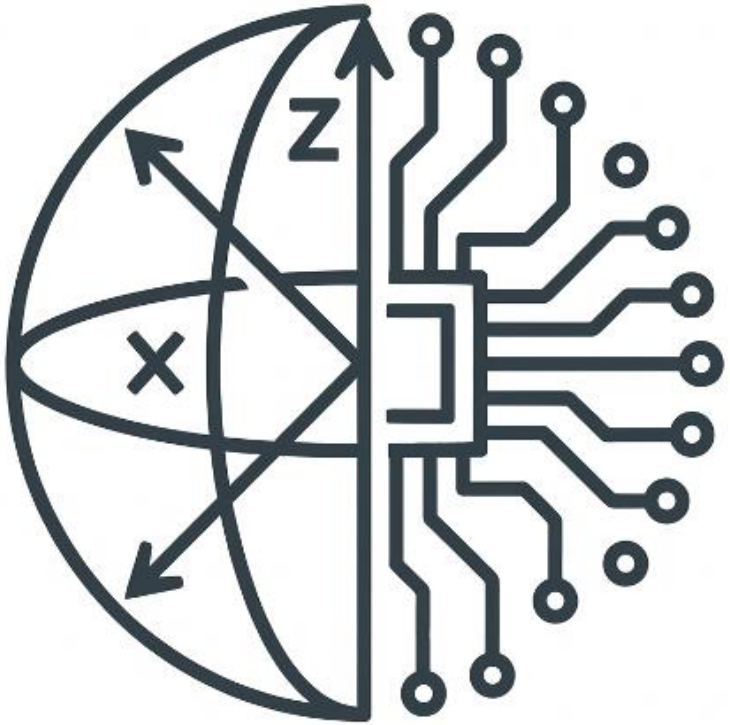
ACM PQQS 2026
MILPITAS, CA, USA

Certified Confinement

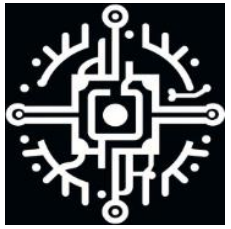
Kernel-Checked Validation of Untrusted Quantum-Cloud Transpiler Output



Ali Hamza Malik
Muhammad Taqi Raza

University of Massachusetts Amherst

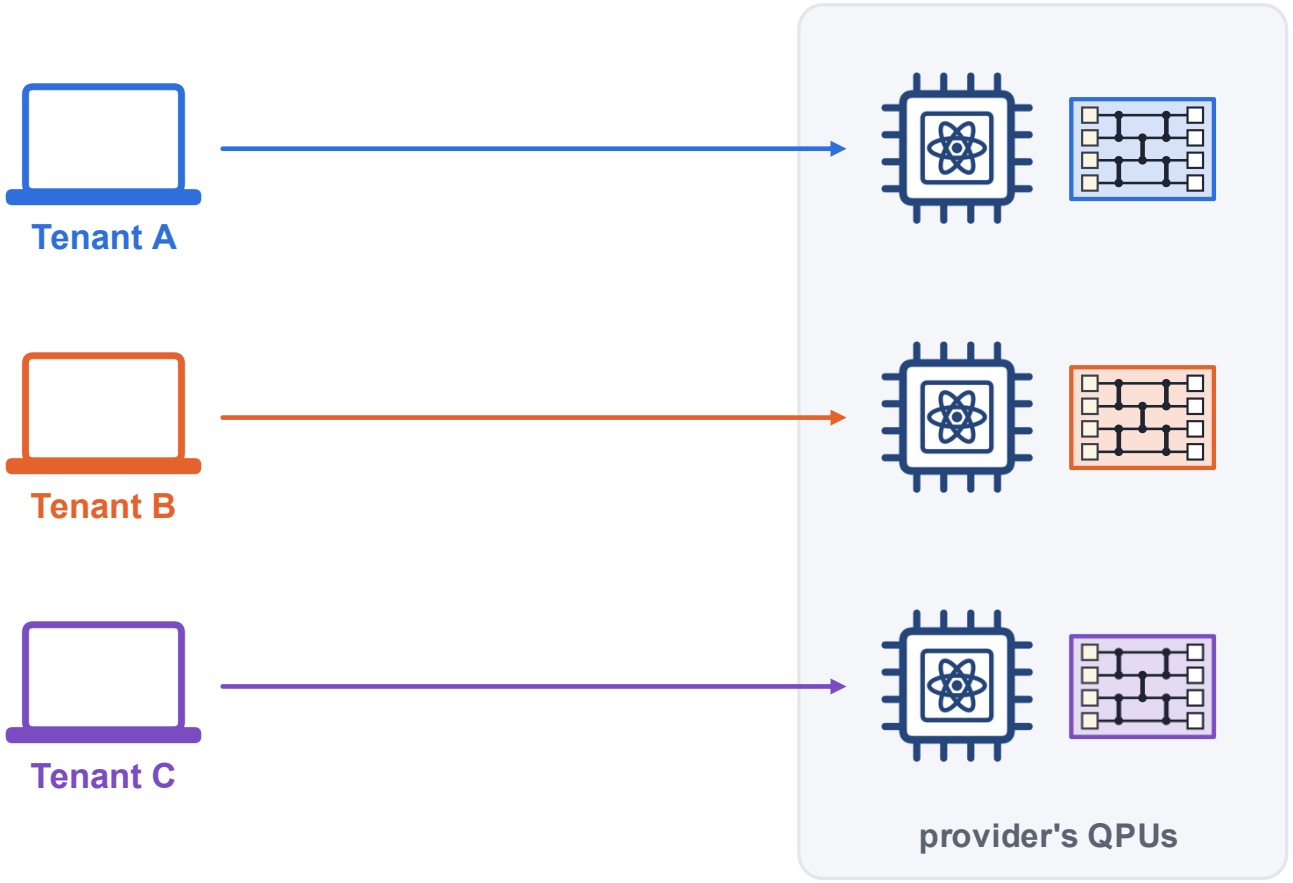
November 2–4, 2026



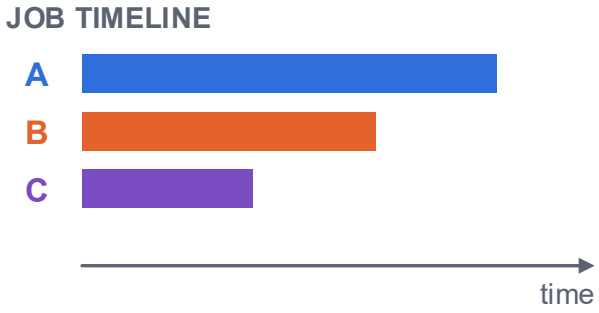
Khwarizmi Lab

University of Massachusetts Amherst

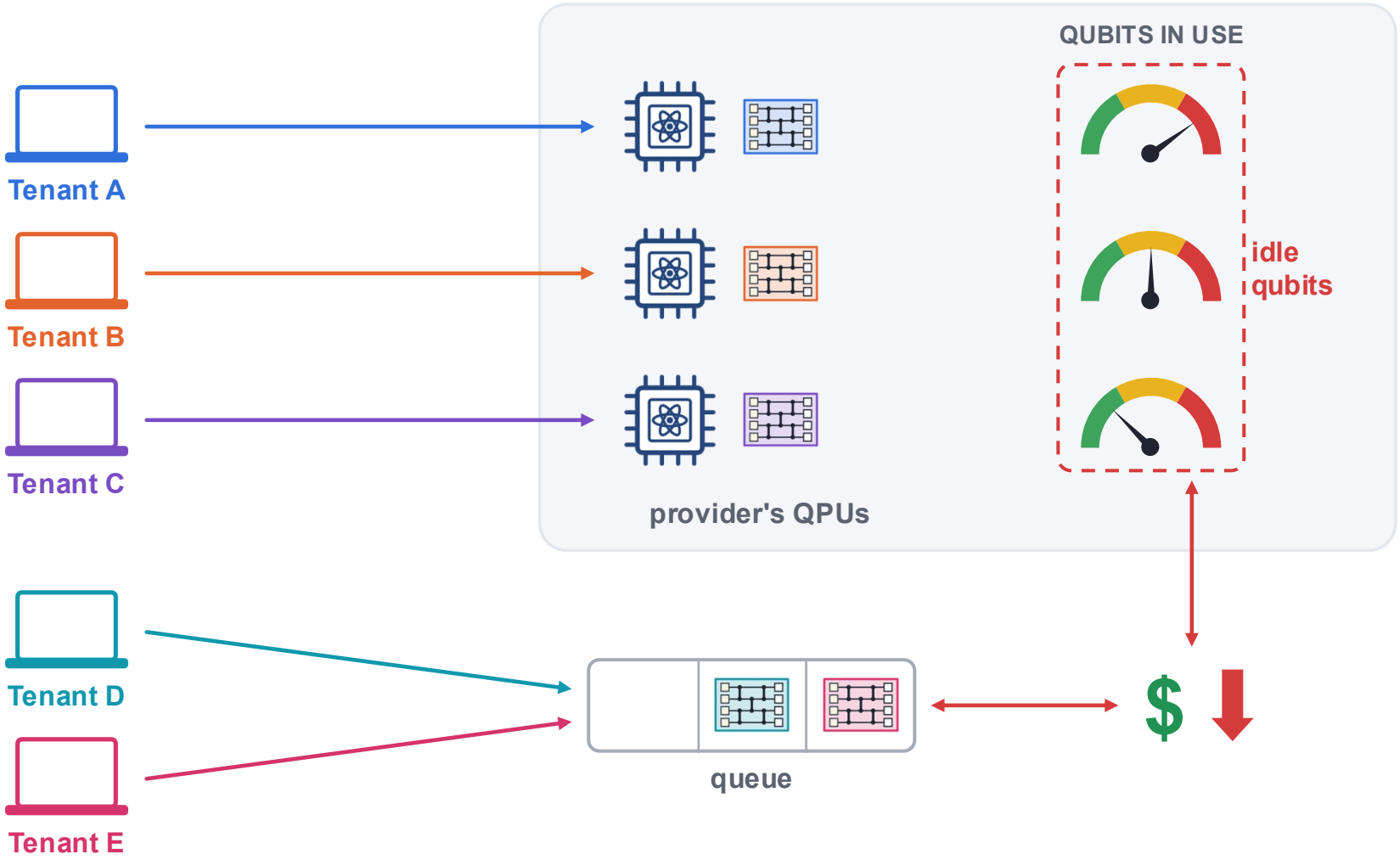
Bare-Bone Quantum Cloud Access



Each tenant gets a whole QPU for a time slot.

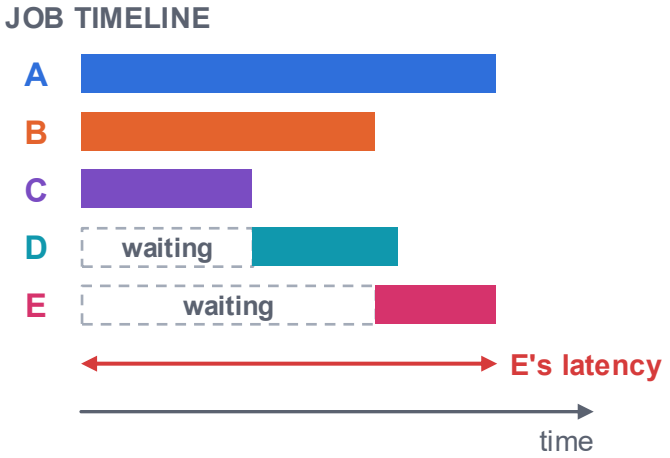


Idle Qubits Increase Tenant Queues

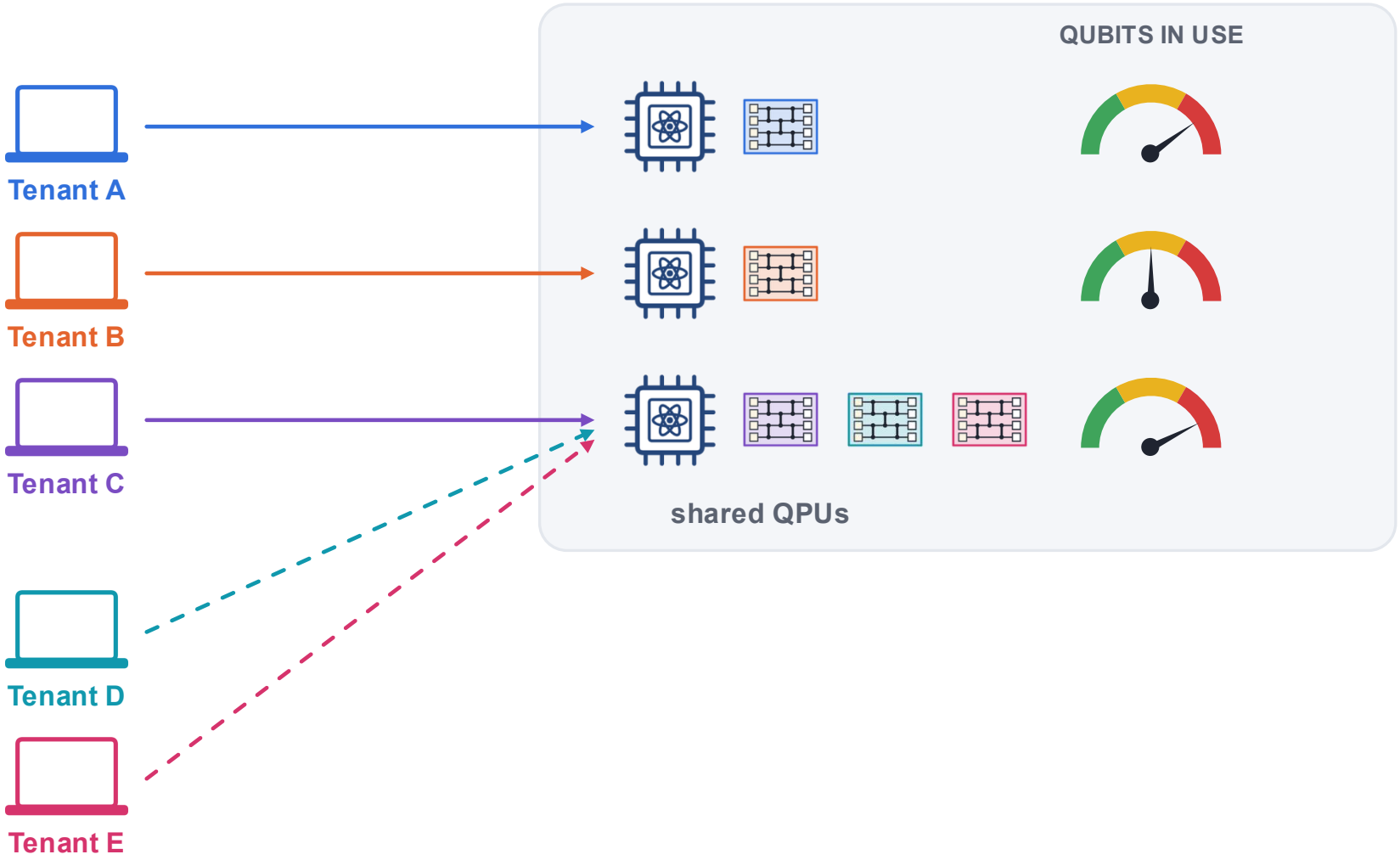


Tenants D and E wait in the queue.

Qubits sit idle while jobs wait.

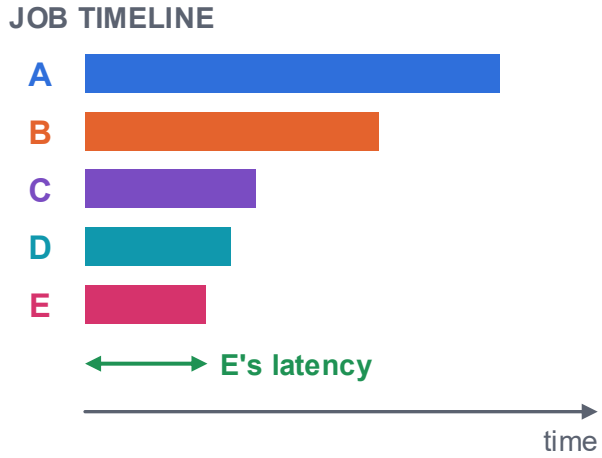


Multi-Tenant Quantum Cloud Sharing

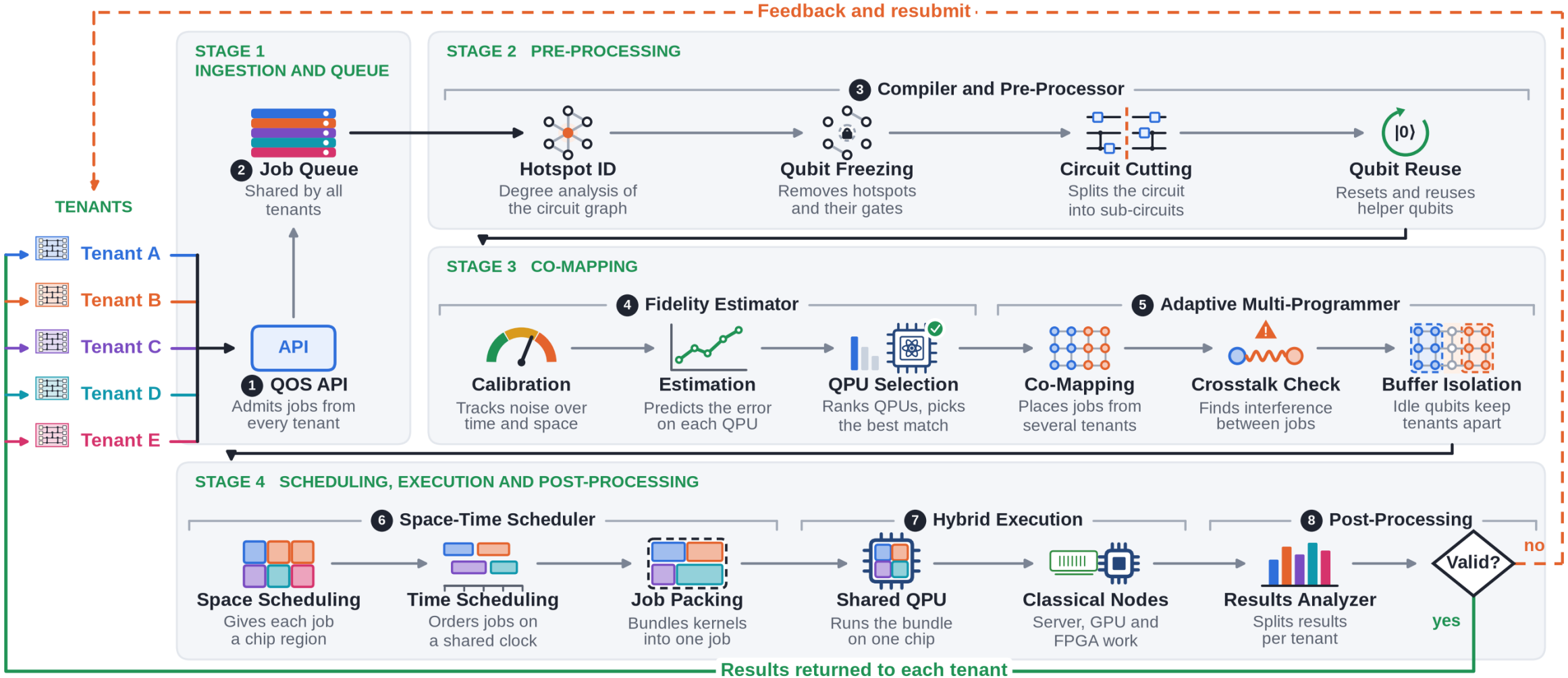


D and E could share QPU 3 with C.

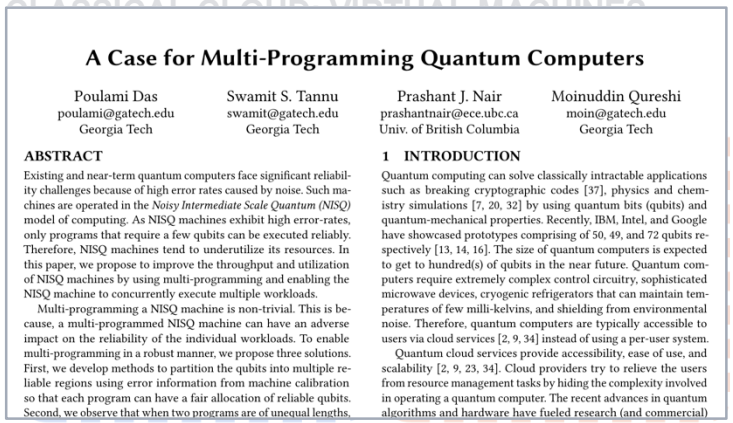
Sharing helps both users and providers.



Multi-Tenant Quantum Cloud Orchestration



Quantum Virtual Machines and Multitasking



A Case for Multi-Programming Quantum Computers

Poulami Das, poulami@gatech.edu, Georgia Tech
Swamit S. Tannu, swamit@gatech.edu, Georgia Tech
Prashant J. Nair, prashantnair@ece.ubc.ca, Univ. of British Columbia
Moinuddin Qureshi, moin@gatech.edu, Georgia Tech

ABSTRACT

Existing and near-term quantum computers face significant reliability challenges because of high error rates caused by noise. Such machines are operated in the *Noisy Intermediate Scale Quantum (NISQ)* model of computing. As NISQ machines exhibit high error-rates, only programs that require a few qubits can be executed reliably. Therefore, NISQ machines tend to underutilize its resources. In this paper, we propose to improve the throughput and utilization of NISQ machines by using multi-programming and enabling the NISQ machine to concurrently execute multiple workloads.

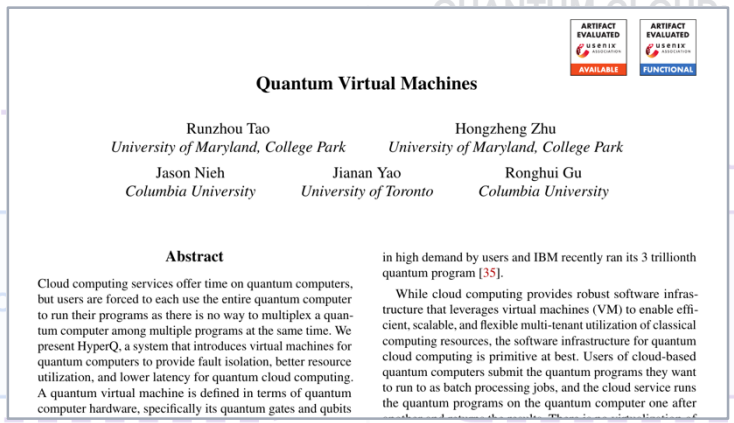
Multi-programming a NISQ machine is non-trivial. This is because, a multi-programmed NISQ machine can have an adverse impact on the reliability of the individual workloads. To enable multi-programming in a robust manner, we propose three solutions. First, we develop methods to partition the qubits into multiple reliable regions using error information from machine calibration so that each program can have a fair allocation of reliable qubits. Second, we observe that when two programs are of unequal lengths,

1 INTRODUCTION

Quantum computing can solve classically intractable applications such as breaking cryptographic codes [37], physics and chemistry simulations [7, 20, 32] by using quantum bits (qubits) and quantum-mechanical properties. Recently, IBM, Intel, and Google have showcased prototypes comprising of 50, 49, and 72 qubits respectively [13, 14, 16]. The size of quantum computers is expected to get to hundred(s) of qubits in the near future. Quantum computers require extremely complex control circuitry, sophisticated microwave devices, cryogenic refrigerators that can maintain temperatures of few milli-kelvins, and shielding from environmental noise. Therefore, quantum computers are typically accessible to users via cloud services [2, 9, 34] instead of using a per-user system.

Quantum cloud services provide accessibility, ease of use, and scalability [2, 9, 23, 34]. Cloud providers try to relieve the users from resource management tasks by hiding the complexity involved in operating a quantum computer. The recent advances in quantum algorithms and hardware have fueled research (and commercial)

MICRO 2019



ARTIFACT EVALUATED usenix AVAILABLE
ARTIFACT EVALUATED usenix FUNCTIONAL

Quantum Virtual Machines

Runzhou Tao, *University of Maryland, College Park*
Hongzheng Zhu, *University of Maryland, College Park*
Jason Nieh, *Columbia University*
Jianan Yao, *University of Toronto*
Ronghui Gu, *Columbia University*

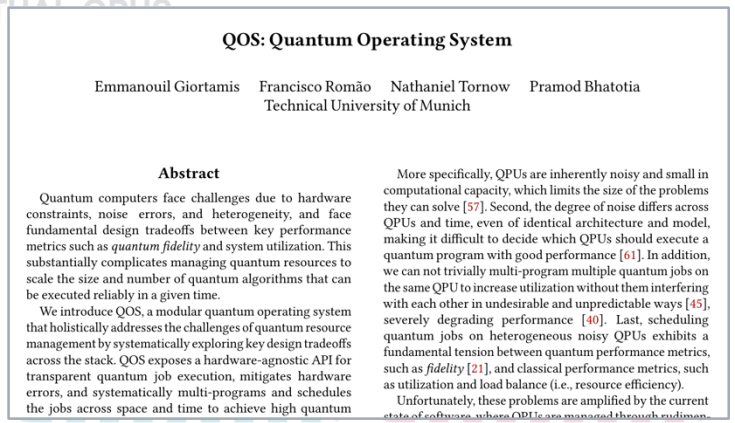
Abstract

Cloud computing services offer time on quantum computers, but users are forced to each use the entire quantum computer to run their programs as there is no way to multiplex a quantum computer among multiple programs at the same time. We present HyperQ, a system that introduces virtual machines for quantum computers to provide fault isolation, better resource utilization, and lower latency for quantum cloud computing. A quantum virtual machine is defined in terms of quantum computer hardware, specifically its quantum gates and qubits

in high demand by users and IBM recently ran its 3 trillionth quantum program [35].

While cloud computing provides robust software infrastructure that leverages virtual machines (VM) to enable efficient, scalable, and flexible multi-tenant utilization of classical computing resources, the software infrastructure for quantum cloud computing is primitive at best. Users of cloud-based quantum computers submit the quantum programs they want to run to as batch processing jobs, and the cloud service runs the quantum programs on the quantum computer one after

OSDI 2025



QOS: Quantum Operating System

Emmanouil Giortamis, Francisco Romão, Nathaniel Tornow, Pramod Bhatotia
Technical University of Munich

Abstract

Quantum computers face challenges due to hardware constraints, noise errors, and heterogeneity, and face fundamental design tradeoffs between key performance metrics such as *quantum fidelity* and system utilization. This substantially complicates managing quantum resources to scale the size and number of quantum algorithms that can be executed reliably in a given time.

We introduce QOS, a modular quantum operating system that holistically addresses the challenges of quantum resource management by systematically exploring key design tradeoffs across the stack. QOS exposes a hardware-agnostic API for transparent quantum job execution, mitigates hardware errors, and systematically multi-programs and schedules the jobs across space and time to achieve high quantum

More specifically, QPUs are inherently noisy and small in computational capacity, which limits the size of the problems they can solve [57]. Second, the degree of noise differs across QPUs and time, even of identical architecture and model, making it difficult to decide which QPUs should execute a quantum program with good performance [61]. In addition, we can not trivially multi-program multiple quantum jobs on the same QPU to increase utilization without them interfering with each other in undesirable and unpredictable ways [45], severely degrading performance [40]. Last, scheduling quantum jobs on heterogeneous noisy QPUs exhibits a fundamental tension between quantum performance metrics, such as *fidelity* [21], and classical performance metrics, such as utilization and load balance (i.e., resource efficiency).

Unfortunately, these problems are amplified by the current

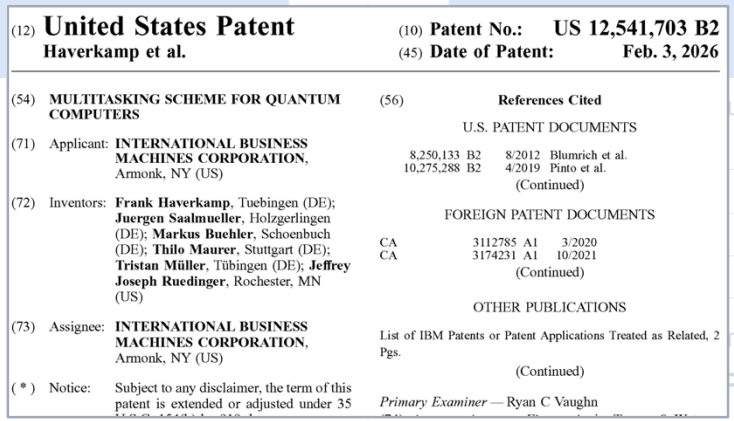
OSDI 2025

Hypervisor

Quantum hypervisor

CPU Memory GPU

Control Readout



(12) **United States Patent**
Haverkamp et al.

(10) **Patent No.: US 12,541,703 B2**
(45) **Date of Patent: Feb. 3, 2026**

(54) **MULTITASKING SCHEME FOR QUANTUM COMPUTERS**

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**, Armonk, NY (US)

(72) Inventors: **Frank Haverkamp**, Tuebingen (DE); **Juergen Saalmueller**, Holzgerlingen (DE); **Markus Buehler**, Schoenbuch (DE); **Thilo Maurer**, Stuttgart (DE); **Tristan Müller**, Tübingen (DE); **Jeffrey Joseph Ruedinger**, Rochester, MN (US)

(73) Assignee: **INTERNATIONAL BUSINESS MACHINES CORPORATION**, Armonk, NY (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35

(56) **References Cited**

U.S. PATENT DOCUMENTS

8,250,133 B2 8/2012 Blumrich et al.
10,275,288 B2 4/2019 Pinto et al.
(Continued)

FOREIGN PATENT DOCUMENTS

CA 3112785 A1 3/2020
CA 3174231 A1 10/2021
(Continued)

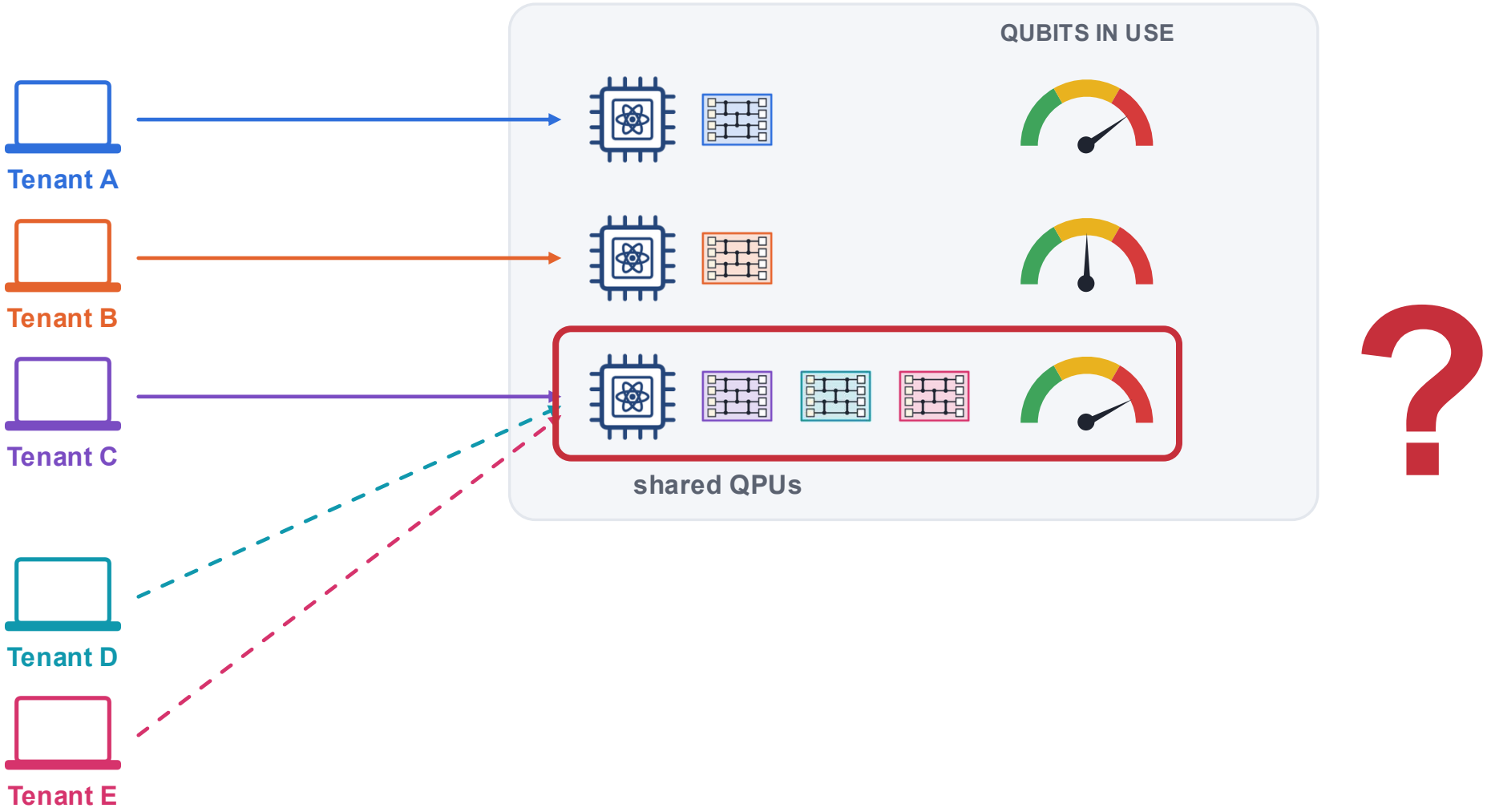
OTHER PUBLICATIONS

List of IBM Patents or Patent Applications Treated as Related, 2 Pgs.
(Continued)

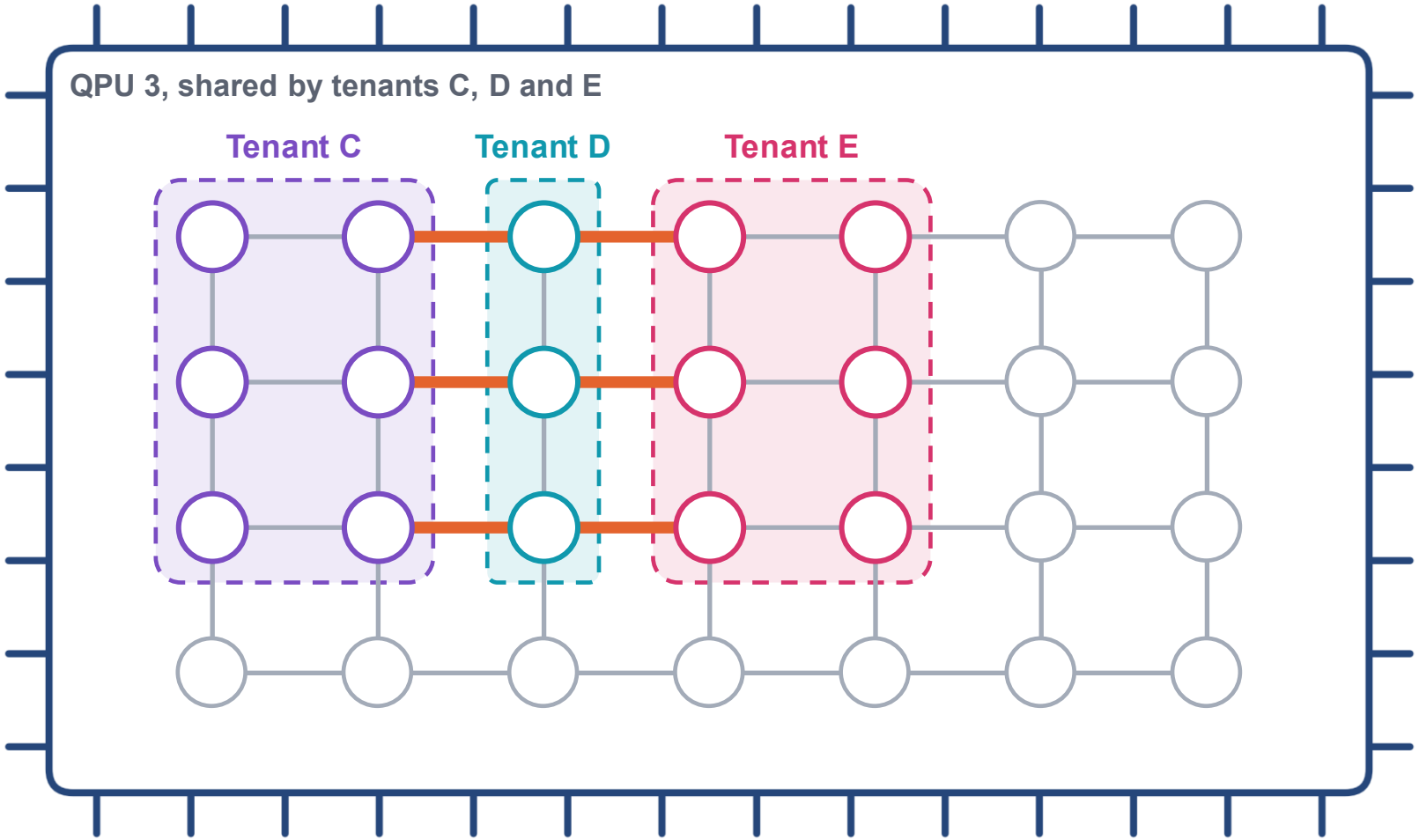
Primary Examiner — Ryan C Vaughn

IBM patent 2026

What Prevents Safe Quantum Cloud Sharing



Tenant Boundaries on Shared Quantum Chips



Tenants' circuits share one chip.

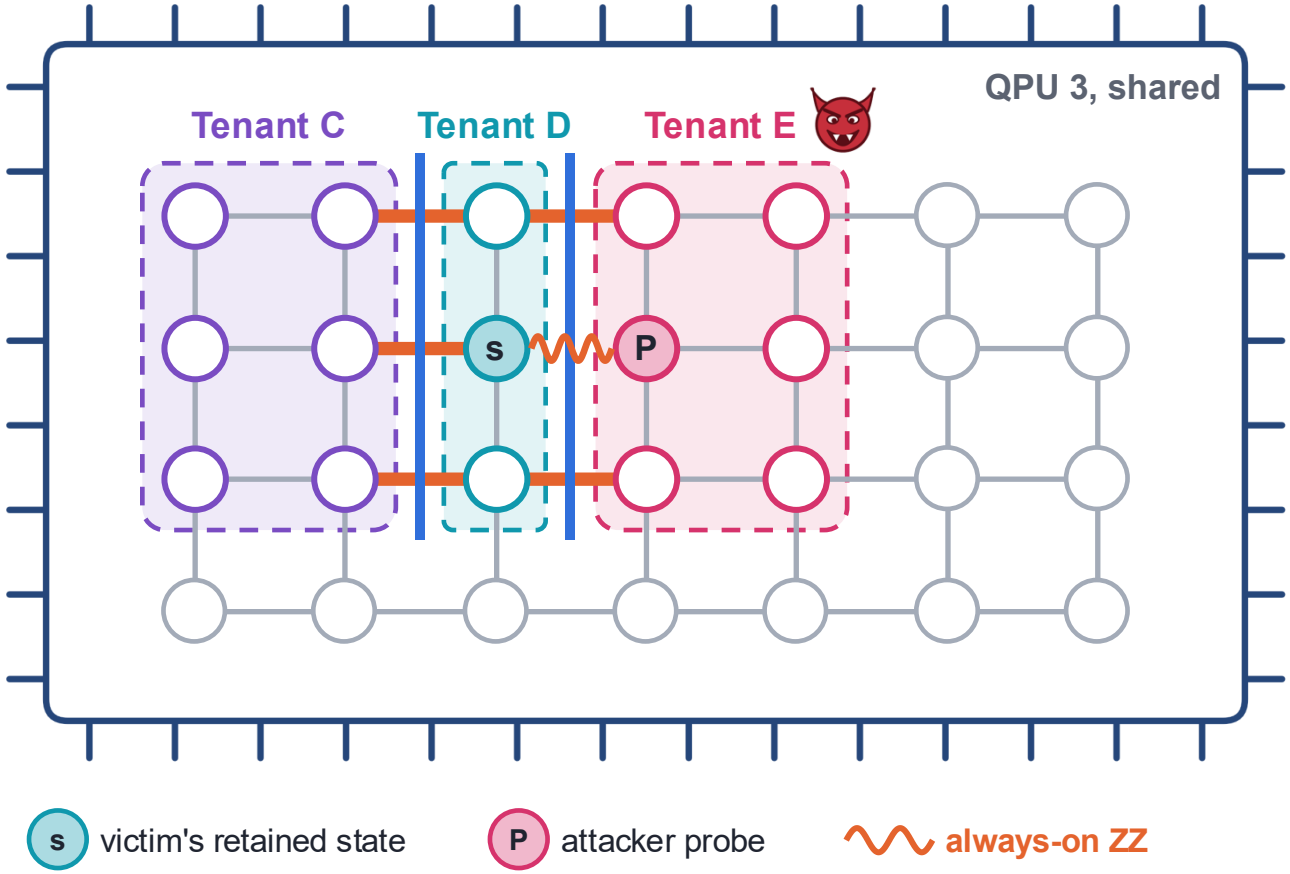
Crosstalk and parasitic coupling act along the edges between tenants.

physical qubit

coupling edge

edge between tenants

Adversarial Threats Across Tenant Boundaries



Passive

Profile a neighbor's circuits and extract its intellectual property.

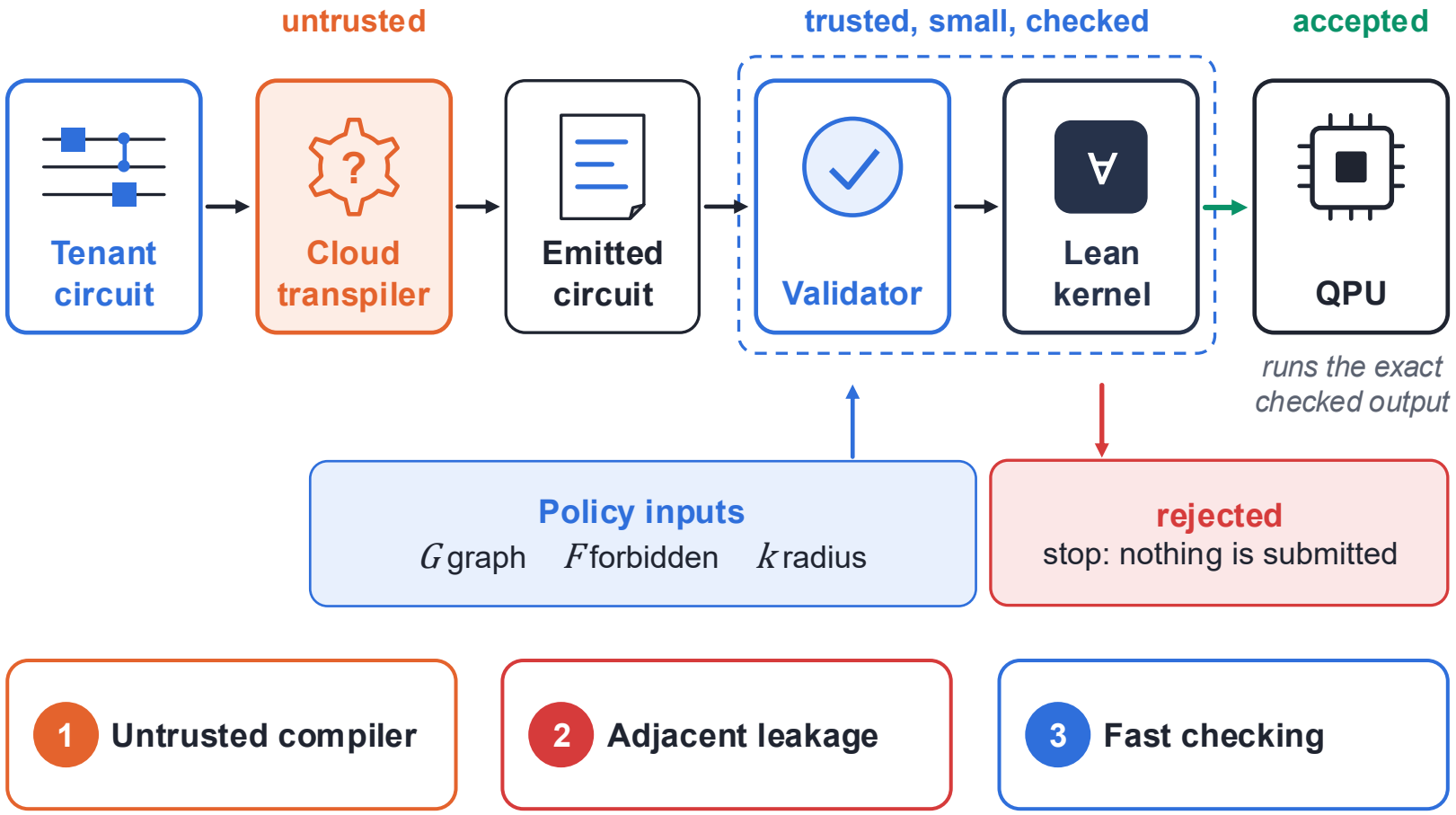
Active

Inject malicious pulses (QubitHammer) to corrupt data.

Impact: confidentiality of proprietary algorithms and integrity of user workloads.

Spatial isolation is the primary security requirement.

Certified Quantum Cloud Validation Pipeline



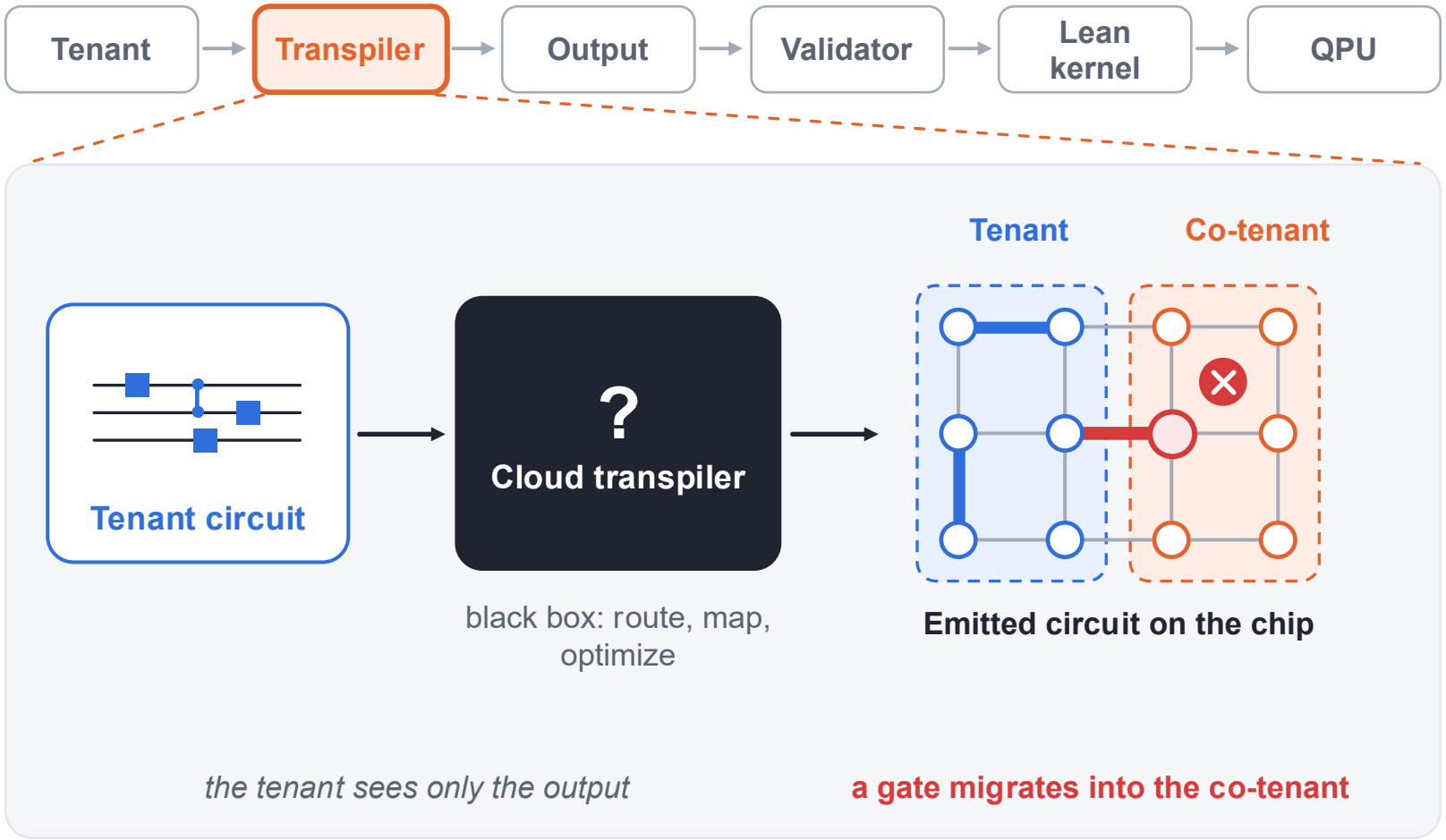
OVERVIEW

Inputs: emitted circuit, coupling graph and forbidden region.

Validation: client-side check, certificate re-checked in Lean.

Check every output before submission.

Untrusted Quantum Cloud Compilation



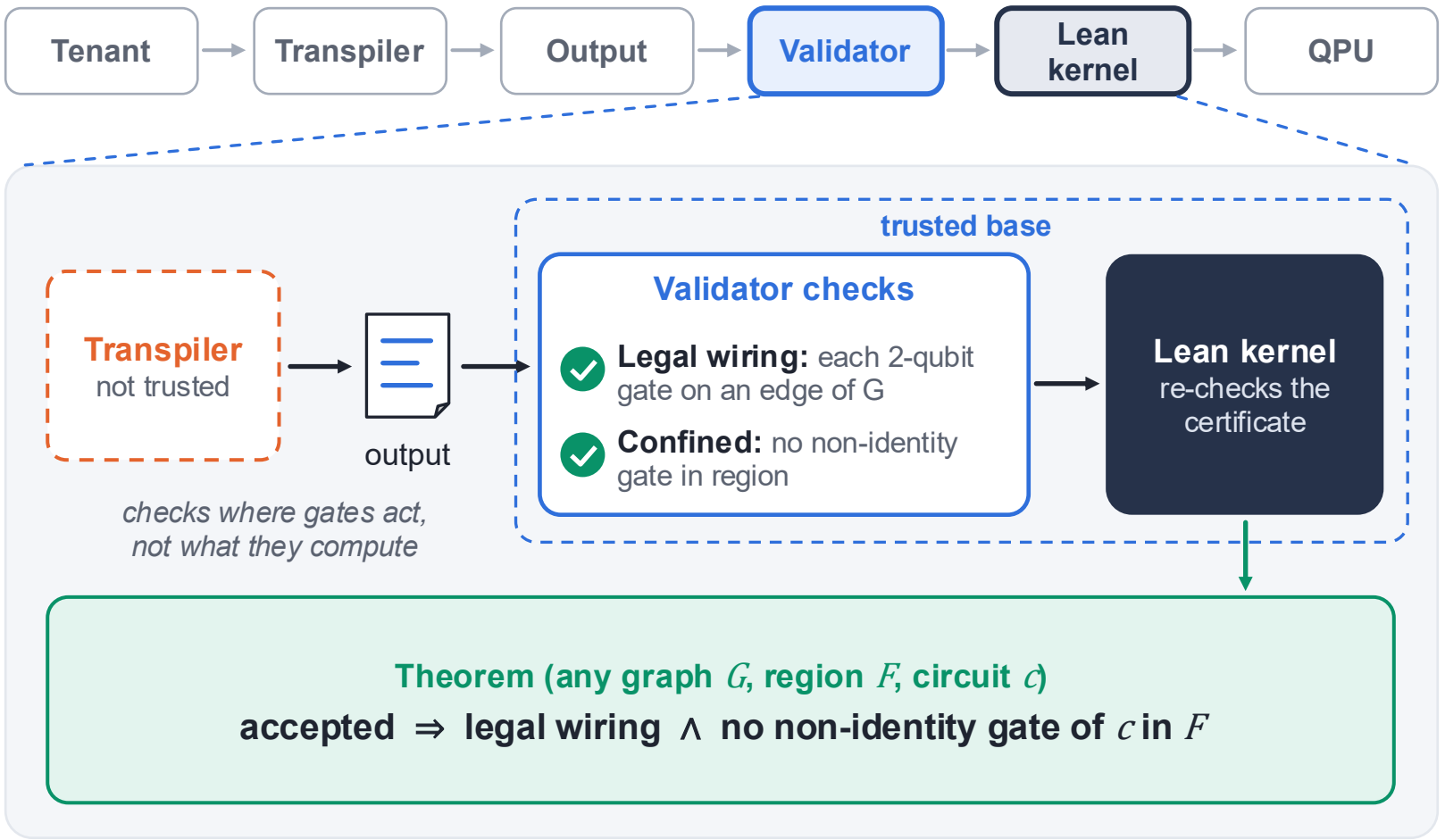
CHALLENGE ONE

Transpiler: places and rewrites gates for the device.

Trust gap: proprietary code may contain bugs or malicious changes.

Who checks the emitted placement?

Kernel-Checked Quantum Circuit Validation



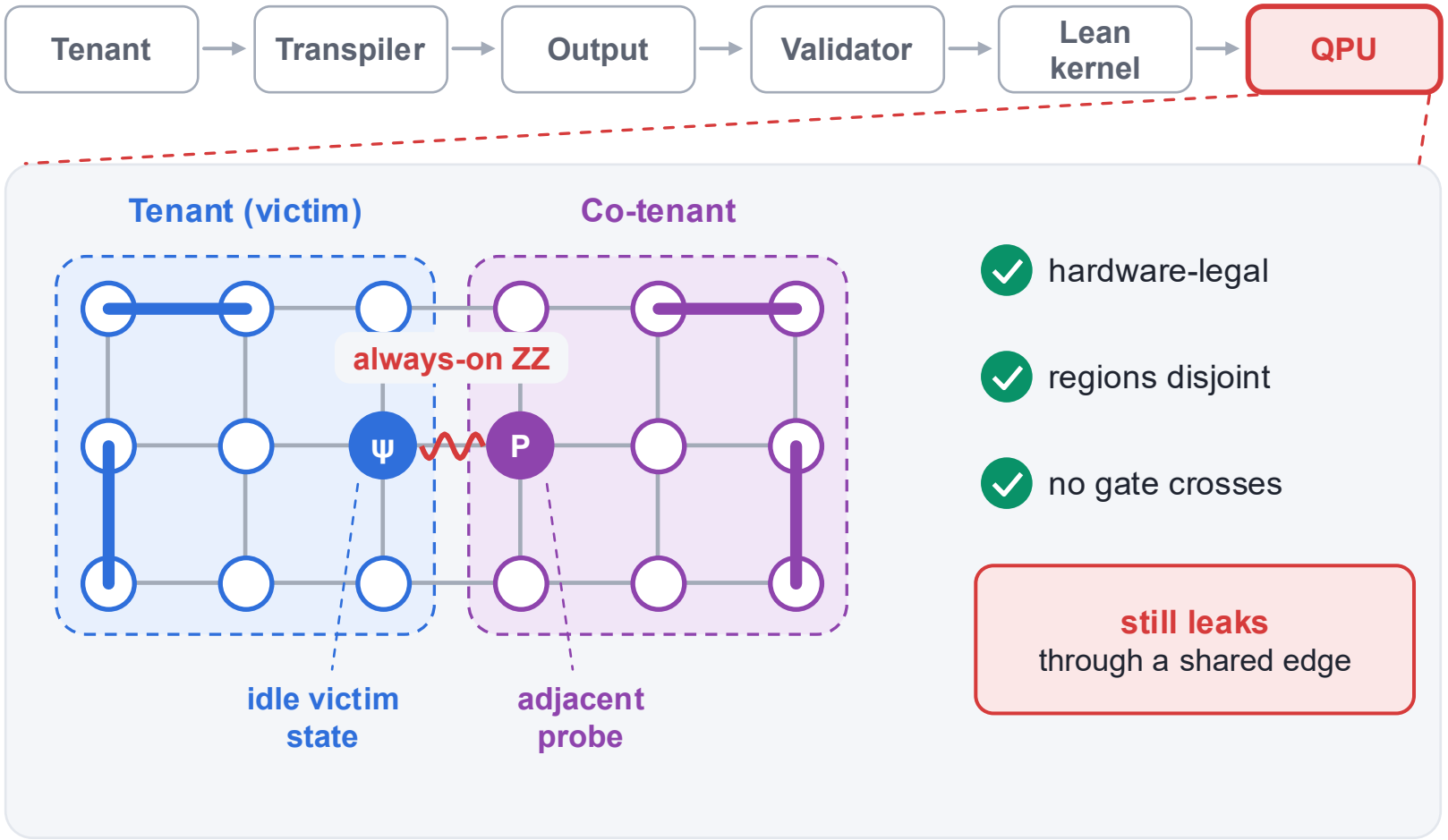
SOLUTION ONE

Translation validation: check each output independently.

Lean kernel: the small core that checks the proof.

The compiler stays outside the trusted base.

Leakage Across Hardware-Legal Tenant Boundaries



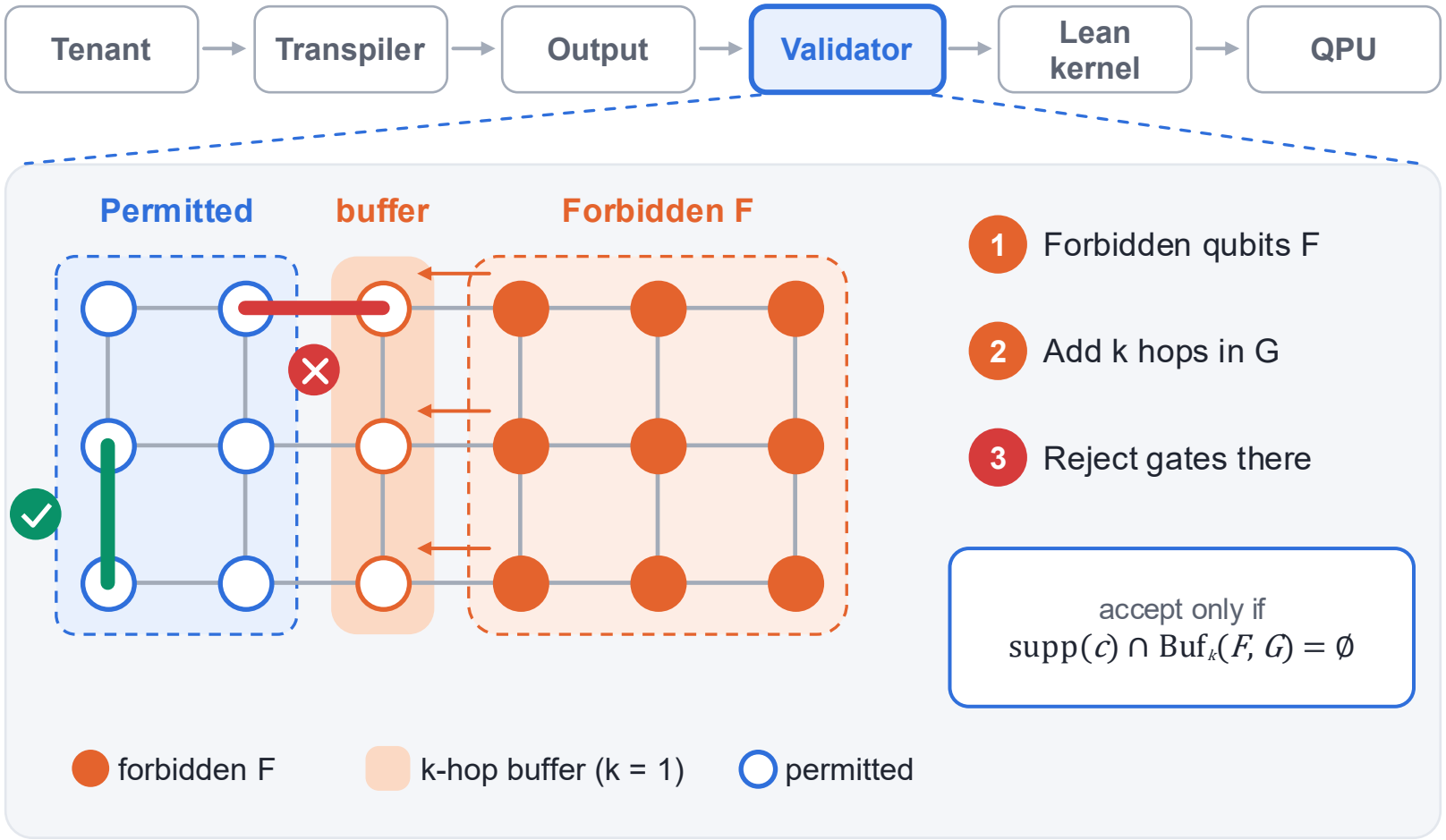
CHALLENGE TWO

Hardware-legal: two-qubit gates follow physical edges.

Always-on coupling: an idle state can affect an adjacent probe.

Disjoint qubit sets are insufficient.

Graph-Based Buffers Separate Tenant Circuits



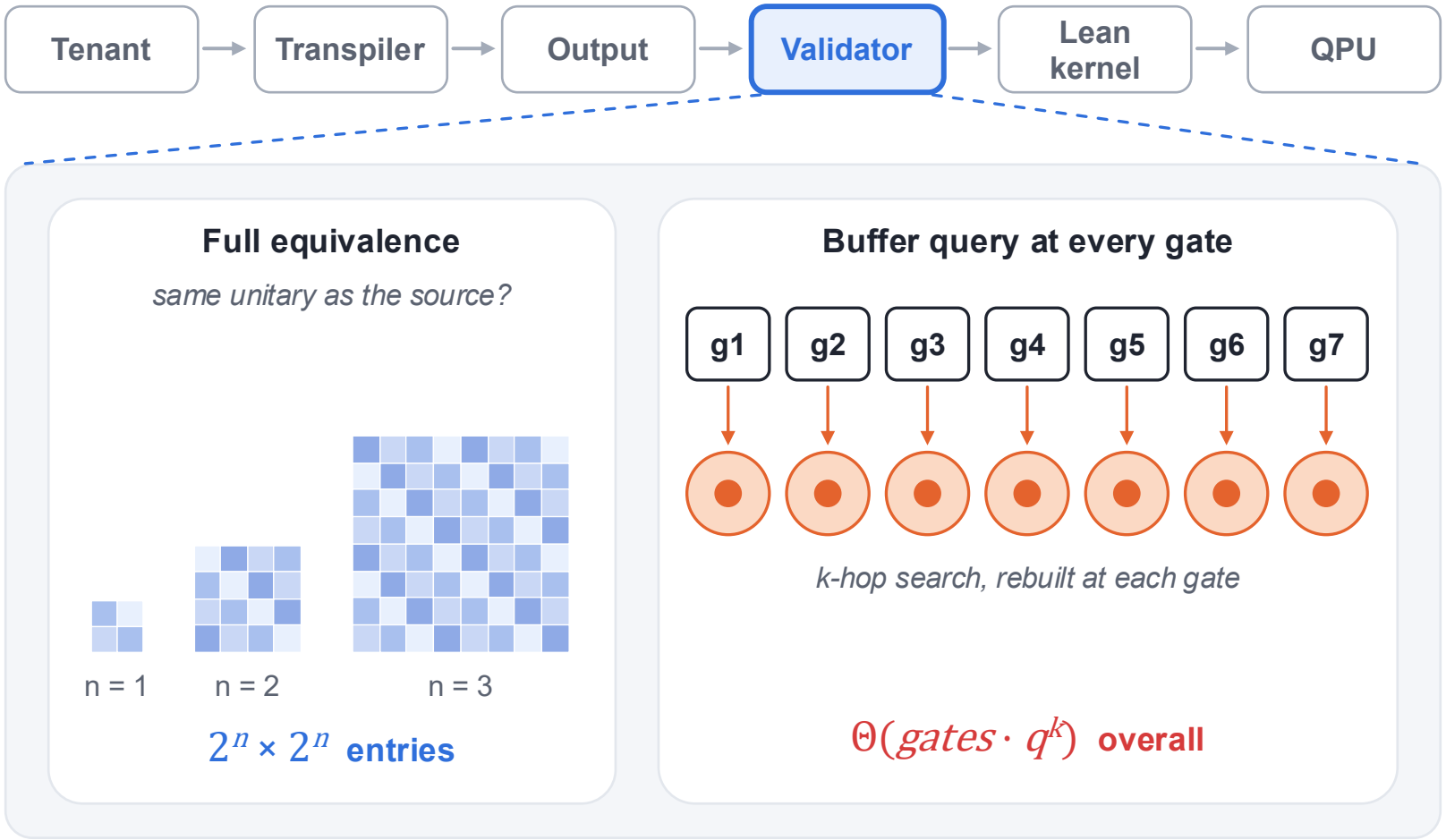
SOLUTION TWO

Forbidden region: co-tenant qubits F.

Buffer: F plus qubits within k coupling-graph hops.

Keep active gates outside the buffer.

Per-Submission Validation Must Be Fast



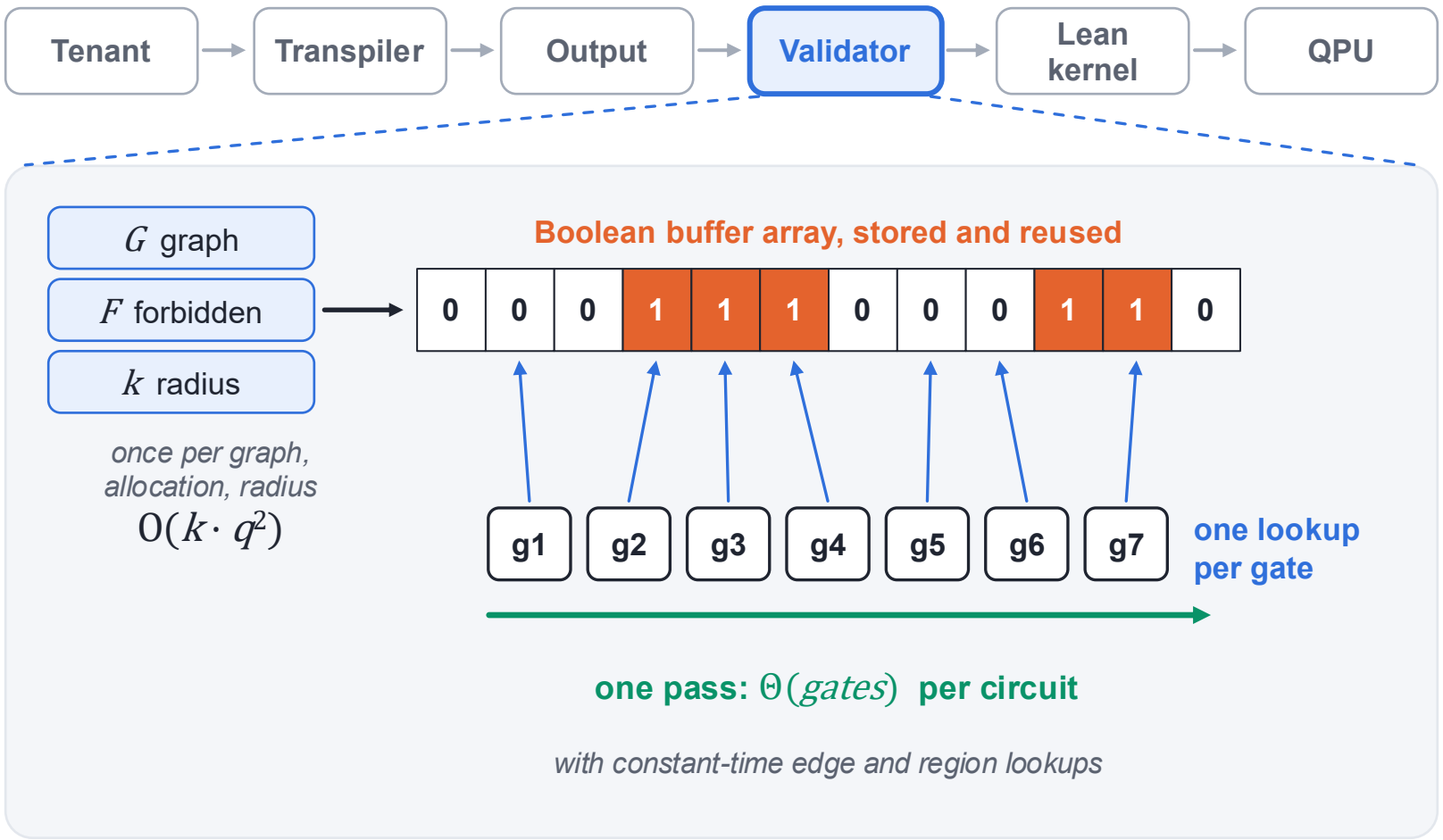
CHALLENGE THREE

Semantic comparison: full unitary matrices grow exponentially.

Repeated buffer queries: Θ(gates · q^k) in the naive check.

Avoid simulation and repeated graph work.

Precomputed Buffers Enable Linear Checking



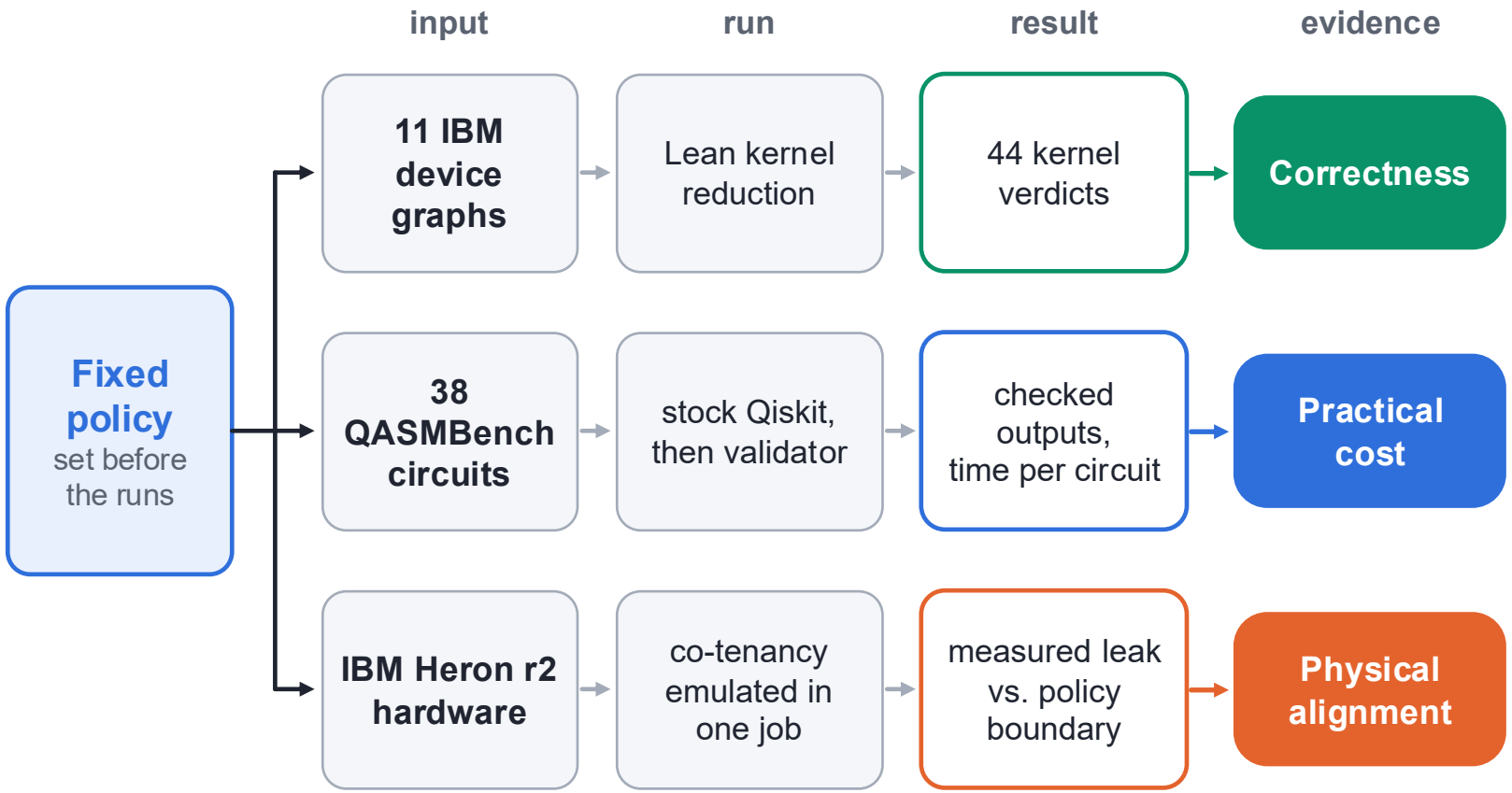
SOLUTION THREE

Precompute: store the buffer as a Boolean array.

Reuse: one structural pass per emitted circuit.

Check where gates act, without simulation.

Three Paths to Evaluation Evidence



EVALUATION

Fixed policy: defined before the hardware campaign.

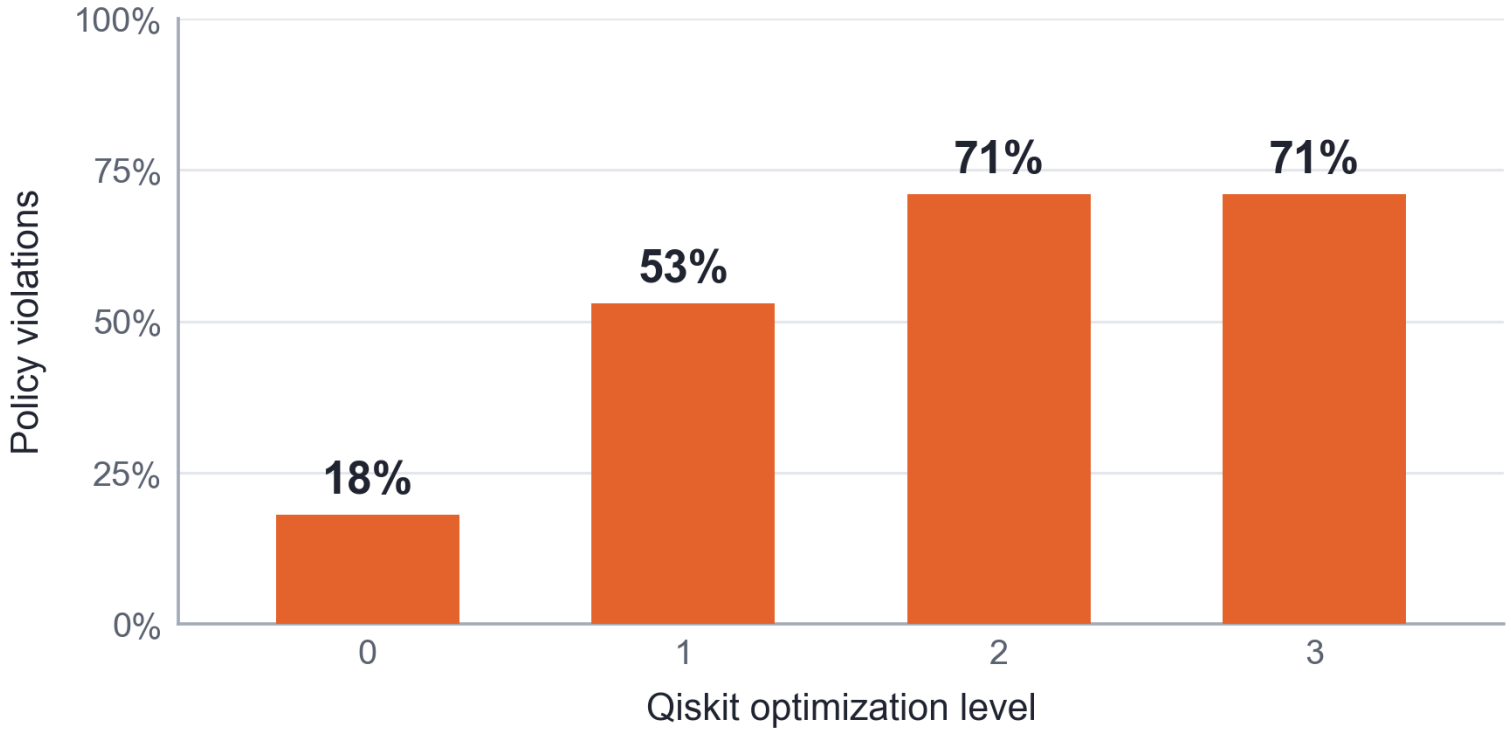
Three arms: graphs, compiler outputs and live hardware.

Test correctness, cost and physical alignment.

Transpiler Optimization Violates Tenant Boundaries

Stock Qiskit, unconstrained placement, 38 QASMBench circuits per level, 12-node ring

every violating output still had legal wiring



Violation: a non-identity gate on a co-tenant qubit or one edge away.

All rejected by the validator.

With the buffer reserved: no false rejections.

The cost: fitting circuits drop from 31 to 23 at one hop.

Circuit Validation Adds Little Overhead

< 1 ms

1,000 gates, native checker

6 ms

10,000 gates, unbuffered, 156-qubit graph

≤ 17%

of benchmark transpile time, usually below 10%

< 5 s

Lean kernel: full 156-qubit device module

Native timing: the checker compiled from Lean.

Kernel evidence: 44 verdicts across 11 IBM graphs, 7–156 qubits.

Practical for per-submission checks.

Hardware Leakage Matches Graph Boundaries

One hop

reject **22/23 detected**



Two or more hops

accept **0/23 detected**



leak detected no leak detected one tile per pair observation

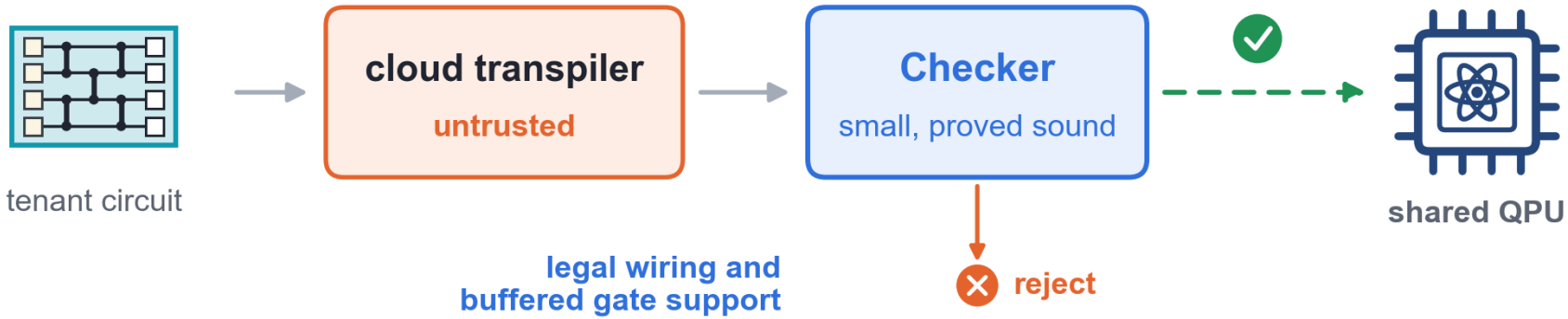
Bands: verdicts from the graph alone, fixed before measuring.

z: the probe's shift in standard errors. A leak is detected when $z \geq 5$.

Cause: always-on ZZ coupling between neighbors.

Every detected leak fell inside the rejected boundary.

Validate Quantum Circuits Before Execution



One pass
A single linear pass, even after routing and optimization.

Proved
Verified in Lean: acceptance ensures confinement.

Tested
Stock Qiskit output and live IBM Heron hardware.

Check untrusted transpiler output before it runs.

Key Takeaway

Scaling quantum clouds needs safe sharing.

A small, verified validator checks that compiler output respects tenant boundaries before execution.

Ali Hamza Malik
University of Massachusetts Amherst

Seeking Internship Opportunities

SCAN TO CONNECT

linkedin.com/in/alihamzamalik

Thank you