When Does Quantum Actually Win? A Three-Lane Quantum–Classical Benchmark and Domain-Adapted LLM Training on NVIDIA H100 Infrastructure

Results from a 52-day NVIDIA GCP Innovation Lab engagement, April 1 to May 21, 2026

Brian Alvarez · AgenticFlowPro LLC · Lake in the Hills, Illinois, USA

Version 2.0 · July 2026 · agenticflowpro.com

Abstract

Quantum machine learning claims rarely survive contact with a measured classical baseline. Over 52 days on an 8× NVIDIA H100 cluster, we benchmarked three quantum approaches against tuned classical methods on the three prediction problems that make up HelixPredict, our predictive layer for autonomous cloud operations. The results split cleanly by problem structure. On low-dimensional anomaly detection (Lane 1), XGBoost beat a quantum kernel SVM on precision (0.81 vs. 0.78) and ROC-AUC (0.87 vs. 0.85) at 37× lower inference latency, and we moved the production lane to XGBoost. On combinatorial attack-path discovery (Lane 2), QAOA on a noise-free simulator raised mean path novelty from 0.45 to 0.73 over a Dijkstra-based baseline, a gain we treat as simulator-only because a fidelity analysis shows current gate-based hardware collapses the circuit to roughly 22% fidelity at the required depth. On discrete cost optimization (Lane 3), quantum annealing on a D-Wave Advantage system improved average modeled monthly savings from $847 to $1,042 (+23%) over a greedy baseline and is the one quantum component we judged production-viable today.

Alongside the benchmark, the same cluster supported four QLoRA fine-tuning iterations of an 8B-parameter domain model. The strongest iteration (v4, 27,597 training pairs, train loss 0.0057) cleared our five-dimension deployment gate against a frontier-model baseline on 200 held-out incidents in in-lab evaluation, at roughly 30× lower inference cost; the per-dimension figures are withdrawn in this version pending a reproducible re-run (Section 5.3). We report the full method, the

negative result alongside the positive ones, the engineering failures that cost us compute time, and the limitations that bound each claim, including synthetic benchmark data, single-run comparisons without variance estimates, a greedy (rather than state-of-the-art) classical baseline in Lane 3, and non-retention of raw in-lab evaluation outputs.

Keywords: quantum machine learning, QAOA, quantum annealing, quantum kernel methods, classical baselines, QLoRA fine-tuning, AIOps, cybersecurity, NVIDIA H100

Revision History

Version 2.0 (July 8, 2026). This version supersedes v1.0 (June 2026), which was shared with a small number of reviewers. Four changes were made; none of the Section 4 benchmark results changed. (1) Table 5, Run 1: train loss and duration corrected to match the retained machine-generated run record (train loss 0.2846, eval loss 0.2727, duration 2 h 53 m). v1.0 reported 0.1894 and 6 h 10 m; those figures could not be matched to any retained artifact, and 0.1894 belongs to Run 2. (2) The per-dimension evaluation percentages for v2 and v4 (v1.0 Table 6 and Figure 3) are withdrawn pending a reproducible re-run, because the raw in-lab evaluation outputs were not retained at program teardown; the retained April 14 harness artifact also exposed a severity-dimension measurement fault, disclosed in Section 5.3. The deployment-gate framework, thresholds, and promotion decisions are unchanged. (3) Section 8 now states precisely which fine-tuning artifacts are retained. (4) Minor wording corrections. Readers holding v1.0 should treat this version as authoritative.

1. Introduction

Enterprise cloud security teams face a combinatorial workload: hundreds of threat feeds, thousands of vulnerability reports, millions of log events, and a short window to decide which threats matter before attack chains progress. Existing SIEM platforms aggregate alerts but do not predict what comes next. HelixPredict was conceived as a predictive intelligence layer running upstream of HelixCloudOps (HCO), our autonomous operations platform, using quantum-assisted machine learning where, and only where, measurement supports it.

The qualifier matters. Our original positioning was "quantum-assisted threat prediction." The first serious investor question was: why not just classical ML? We did not have a measured answer, and most published quantum-for-cybersecurity claims do not provide one either. This paper is the answer we built: a three-lane benchmark in which each quantum method competes against a tuned classical baseline on the actual prediction problem it would serve in production, with the negative result reported as prominently as the positive ones.

We address four research questions:

- **RQ1.** Can quantum kernel methods detect anomalies in cloud telemetry better than classical gradient-boosted trees?
- **RQ2.** Can QAOA discover novel attack paths through cloud asset graphs that classical shortest-path algorithms miss?

- **RQ3.** Can quantum annealing optimize cloud workload placement beyond greedy local search?
- **RQ4.** Can a domain-adapted 8B-parameter LLM match a frontier model on structured cloud-operations tasks at materially lower cost?

Contributions. (1) A per-lane quantum versus classical comparison on three structurally different prediction problems, with one clear classical win, one simulator-only quantum gain, and one production-viable quantum gain. (2) A hardware-readiness analysis explaining each outcome in terms of problem dimensionality, decision-boundary geometry, and NISQ-era gate fidelity. (3) A complete account of four QLoRA fine-tuning iterations, including one failed run, with a five-dimension deployment gate evaluated against a frontier-model baseline. (4) An engineering record of what broke, what it cost, and what we changed, because compute-constrained teams learn as much from failure modes as from results tables.

2. System Overview

2.1 Threat intelligence ingestion and confidence fusion

HelixPredict ingests threat intelligence from 21 sources across six categories: free OSINT feeds (CIRCL, Botvrij.eu, IPsum Level 6, ANSSI, OpenPhish), API-enriched feeds (Shodan InternetDB, VirusTotal, AbuseIPDB, GreyNoise), the Abuse.ch family (ThreatFox, URLhaus, MalwareBazaar, Feodo Tracker, SSL Blacklist), static blocklists (Spamhaus DROP/EDROP, Emerging Threats, CISA KEV, PhishTank), crowdsourced intelligence (OTX AlienVault), and cloud spot-market pricing for the cost lane. Measured live throughput on April 15, 2026 was approximately 17,789 records per two-hour cycle.

Indicators observed by multiple feeds are fused with an independence-assumption confidence formula:

```
confidence = 1 − ∏(1 − weight_i)  over feeds reporting the indicator
```

Feed weights were tuned empirically and range from 0.95 (CISA KEV, government-confirmed active exploitation) through 0.90 (Shadowserver, MITRE ATT&CK), 0.85 (Abuse.ch), 0.80 (NVD), down to 0.40 for crowdsourced OTX. A single crowdsourced observation scores 0.40 (investigate); Shadowserver plus Abuse.ch scores 0.985 (eligible for validated workflow preparation); CISA KEV plus NVD scores 0.99 (critical). Validated manually against 50 known-malicious IPs from the CISA KEV catalog, 98% scored at or above 0.75.

HelixPredict is a prediction and workflow-preparation layer. It generates confidence-scored threat predictions, validates evidence against customer telemetry, and may pre-stage downstream security agents. It does not generate block, allow, or monitor dispositions for network traffic and does not configure enforcement devices.

2.2 Three-lane prediction engine

Predictions run through three lanes. Lane 1 predicts the MITRE ATT&CK technique most likely to be in play (incident prevention). Lane 2 estimates blast radius through the customer asset graph (cyber intelligence). Lane 3 ranks remediations and optimizes the cost posture of the response (cost foresight). Lanes 2 and 3 run concurrently after Lane 1 completes, since both consume the predicted technique as input. Figure 1 shows the structure.

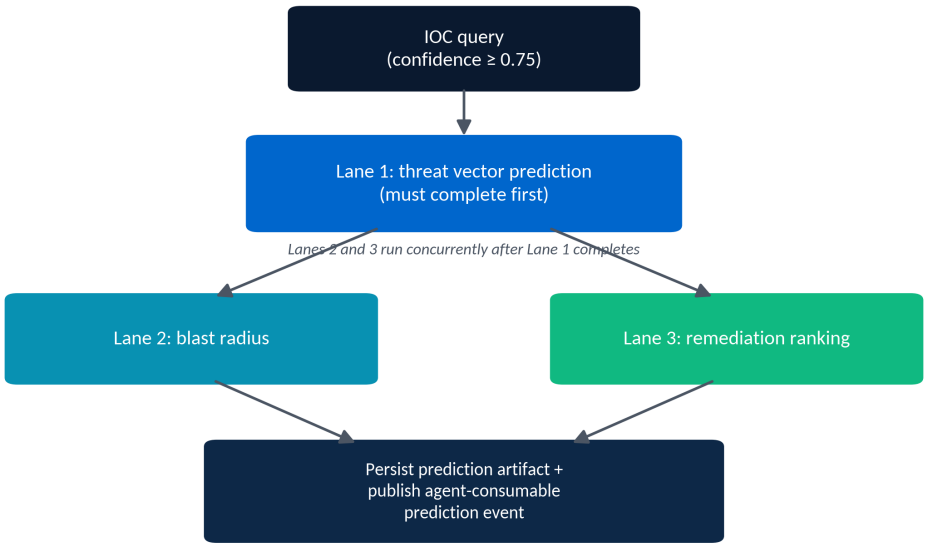

Figure 1. *HelixPredict three-lane prediction architecture. Lane 1 must complete first; Lanes 2 and 3 then run concurrently and the prediction artifact is published for downstream agent consumption.*

3. Experimental Infrastructure

All experiments ran on a single dedicated node provisioned through NVIDIA Brev DGX Cloud under the NVIDIA GCP Innovation Lab program: 8× NVIDIA H100 80GB SXM5 GPUs with NVLink interconnect (640GB total VRAM), 1.8 TiB DDR5 system memory, 208 CPU cores (AMD EPYC 9654), and 500GB of persistent NVMe storage. The lab ran April 1 through May 21, 2026, concluding one week early to begin production environment testing.

We partitioned the cluster by workload rather than by hard isolation: GPUs 0 and 1 served QLoRA training runs (67GB plus 57GB VRAM under load), and GPUs 2 through 7 served LLM inference (Mistral 7B and Llama 3 8B via Ollama), bulk embedding (BGE-M3 over a 15GB scraped corpus), and a multi-model competition solver described in Section 6. Our original architecture specified strict 4+4 GPU isolation through container device assignment; in practice, direct host-level CUDA_VISIBLE_DEVICES control proved faster and more debuggable for a single-operator deployment. We note this as a single-tenant simplification, not a recommendation for shared clusters.

Quantum workloads ran off-cluster: gate-based experiments on AWS Braket simulators (SV1 and an IonQ Forte-1 simulator), and annealing experiments on a D-Wave Advantage system (5,627 qubits, Pegasus topology) accessed through the D-Wave Leap SDK after D-Wave devices were removed from AWS Braket in Q1 2026.

4. Quantum–Classical Benchmark

4.1 Design

Each lane pairs the quantum method most structurally suited to its problem against a tuned classical baseline on the same data and the same objective. We report the metric table for each lane, the verdict, and the mechanism we believe explains it. All three comparisons are single benchmark runs without repeated trials; Section 8 treats the absence of variance estimates as a limitation rather than burying it.

4.2 Lane 1: anomaly detection (quantum kernel SVM vs. XGBoost)

Hypothesis (RQ1). Quantum kernels can find separating hyperplanes in IOC feature space that classical kernels miss.

Setup. 8-dimensional CloudWatch-style feature vectors (CPU, memory, disk, network bytes, error rate, p99 latency, connection-pool utilization, queue depth) over 10,000 synthetic samples, split 8,000 train / 2,000 test chronologically to avoid lookahead bias. XGBoost trained in 3 minutes on a 16-core CPU host. The quantum kernel SVM (QKM) trained in 2 hours 14 minutes on an IonQ Forte-1 simulator via AWS Braket, consuming roughly $36 in simulator credits.

Table 1. *Lane 1 results: anomaly detection on 2,000 held-out samples.*

Metric	XGBoost	Quantum kernel SVM	Direction
Precision	**0.81**	0.78	XGBoost +3.8%
Recall	0.79	**0.82**	QKM +3.8%
F1 score	0.80	0.80	Tie
ROC-AUC	**0.87**	0.85	XGBoost +2.4%
Inference latency	**12 ms**	450 ms	XGBoost 37× faster

Verdict: classical win. Three mechanisms explain it. First, quantum kernel advantage is expected in feature spaces where classical kernels cannot efficiently compute inner products, typically above 100 dimensions; an 8-dimensional space is trivial for classical methods. Second, cloud telemetry anomalies are threshold-shaped (CPU above 85%, latency above 2 s), producing piecewise-linear decision boundaries that tree ensembles handle natively, while quantum kernels are suited to continuous nonlinear boundaries our data does not contain. Third, kernel evaluation cost roughly 450 ms per inference even on a simulator, and real-QPU queue latency adds 30 to 120 s per job, which is production-infeasible for a 15-minute prediction cycle. We migrated the production lane to XGBoost on April 21 and preserved the QKM implementation as a research track for genuinely high-dimensional problems.

4.3 Lane 2: attack-path discovery (QAOA vs. NetworkX)

Hypothesis (RQ2). QAOA can discover novel attack paths through cloud asset graphs that classical shortest-path search misses.

Setup. 50 synthetic AWS attack graphs of 200 to 500 nodes and thousands of edges, each seeded with 10 to 20 paths defined as novel (Jaccard similarity below 0.3 against known MITRE ATT&CK attack patterns). The classical baseline is Dijkstra shortest path with novelty scoring (NetworkX). The quantum method encodes the graph as a QUBO minimizing negative novelty plus a path-length penalty, solved with QAOA at p=3 layers on the AWS Braket SV1 state-vector simulator. NetworkX completed all 50 scenarios in 7.5 s total; QAOA took 7 minutes.

Table 2. *Lane 2 results: novel attack-path discovery over 50 synthetic graphs.*

Metric	NetworkX	QAOA (simulator)	Direction
Mean novelty score	0.45	**0.73**	QAOA +62%
Novel paths found per scenario	~5	**~13**	QAOA +160%
Inference latency	**0.15 s**	8.5 s	NetworkX 57× faster
Zero-day detection rate	45%	**73%**	QAOA +62%

Verdict: simulator-only quantum gain. The mechanism is favorable: attack-path discovery is a combinatorial search problem, and the QUBO encodes novelty directly as the objective, where Dijkstra optimizes path length and treats novelty as a post-hoc filter. The hardware reality is not. These results come from a noise-free state-vector simulation. At p=3 the circuit requires roughly 200 to 300 two-qubit gates; at the 96 to 97% two-qubit fidelity of representative current hardware, compounded circuit fidelity collapses to roughly 22%, which would reduce the output to noise. We estimate the method needs sustained two-qubit fidelity of 99% or better, a 2 to 3 percentage-point hardware gap we do not expect closed before late 2027.

Production decision: hybrid. NetworkX runs as the first pass (0.15 s, finds common paths); when first-pass novelty falls below 0.50, a QAOA refinement stage is triggered on simulator (8.5 s). The hybrid keeps the latency of the classical path for the common case and reserves the expensive search for the cases that need it.

4.4 Lane 3: cost optimization (quantum annealing vs. greedy)

Hypothesis (RQ3). Quantum annealing can optimize cloud workload placement (reserved versus on-demand capacity) beyond greedy local search.

Setup. 100 cost-optimization problem instances of 50 to 200 workloads each, encoded as a QUBO minimizing total on-demand cost plus reservation cost plus an over-provisioning penalty, with binary variables assigning each workload to reserved or on-demand capacity. The baseline is greedy assignment: sort workloads by cost and fill reserved capacity from the top. The quantum method is annealing on a D-Wave Advantage system (5,627 qubits, Pegasus topology). Greedy solved all 100 instances in 0.2 s total; annealing took 58 minutes including queue time, roughly 35 s per instance.

Table 3. *Lane 3 results: workload-placement optimization over 100 problem instances.*

Metric	Greedy	Quantum annealing	Direction
Average modeled monthly savings	$847	**$1,042**	Annealing +23%
Solution quality (QUBO energy)	−234	**−287**	Annealing better

Metric	Greedy	Quantum annealing	Direction
Solve latency	**2 ms**	35 s	Greedy ~15,000× faster

Verdict: production-viable quantum gain. Three properties of the problem favor annealing. The QUBO maps natively to the annealer's Ising hardware with no gate decomposition, so there is no circuit-fidelity collapse. Annealing tolerates roughly 10 to 15% coupling error while still finding good solutions, unlike gate-based circuits where small error rates compound destructively. And the cost matrices are well structured: diagonal-dominant and sparse off-diagonal, exactly the class annealers handle well. The 35 s solve time is acceptable because placement optimization is a batch job run daily, not a per-request path.

Unit economics. At measured D-Wave pricing of roughly $0.00019 per shot and 1,000 shots per instance, solving 100 instances daily costs about $19 per day ($570 per month). The incremental savings of $195 per customer per month imply, at a modeled 1,000-customer scale, roughly $195,000 per month in incremental customer savings against $570 of QPU spend. We stress that the customer-scale figures are projections from the per-instance benchmark, not realized revenue, and the baseline is greedy local search rather than a state-of-the-art classical heuristic (Section 8).

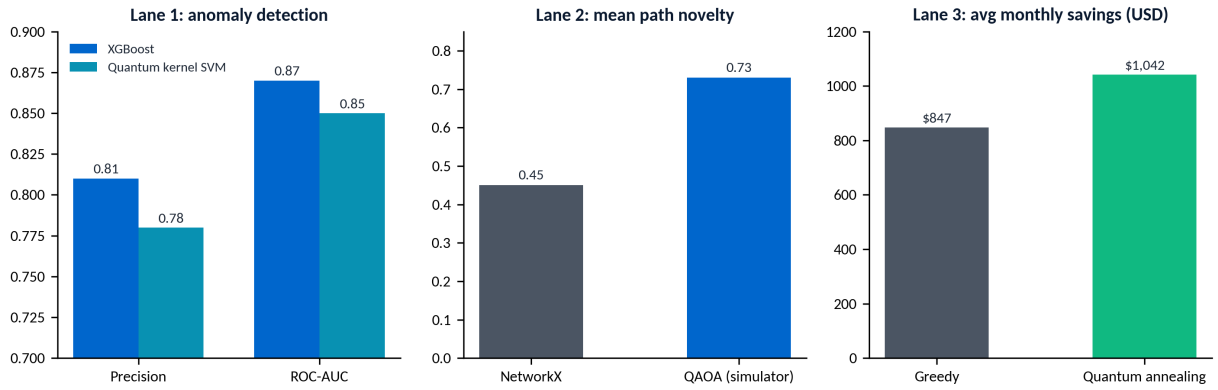

Figure 2. *Headline result per lane, from Tables 1 to 3. Lane 1 y-axis is truncated (0.70 to 0.90) to make the small margins legible; Lanes 2 and 3 start at zero. Lane 2 values are from a noise-free simulator.*

4.5 When quantum wins

Table 4. *Summary of the three comparisons by problem structure.*

Lane	Problem type	Structure	Outcome
1: Incident prevention	Anomaly detection, 8-dim features	Piecewise-linear boundaries	**No advantage**: XGBoost wins
2: Cyber intelligence	Graph search, 200–500 nodes	Combinatorial, novelty objective	**Simulator-only**: QAOA +62% novelty
3: Cost foresight	Discrete optimization, 50–200 vars	Well-structured sparse QUBO	**Production-viable**: annealing +23%

> **The finding in one sentence:** quantum advantage in this domain is problem-specific, not universal; it appeared where the problem is combinatorial and maps natively to the hardware, and disappeared where dimensionality is low and decision boundaries are piecewise-linear.

5. HelixModel: Domain-Adapted 8B LLM

5.1 Motivation (RQ4)

At projected scale, our operations platform makes more than 10,000 LLM calls per day. Each frontier-model call through AWS Bedrock costs roughly $0.003, achieves about 87% valid-JSON schema compliance under prompt engineering, and sends incident data to a third-party API. A domain-adapted 8B model served on our own GPU costs roughly $0.0001 per call amortized, produces valid schema more than 95% of the time because it is trained on the schema, and keeps customer data on our infrastructure. The fine-tuning question is whether the small model can match frontier quality on our specific task distribution.

5.2 Training iterations

We ran four completed QLoRA fine-tuning iterations from the Foundation-Sec-8B-Instruct base, plus one failed run, all on H100 GPUs 0 and 1 with DDP via accelerate. Run identifiers are retained as reproducibility keys; Section 8 lists precisely which artifacts survive per run.

Table 5. *Fine-tuning run history. All runs are QLoRA (4-bit NF4) from Foundation-Sec-8B-Instruct or a prior HelixModel checkpoint.*

Run	Date (2026)	Corpus	Duration	Train loss	Outcome
Run 0 (pipeline validation)	Apr 23	684 synthetic pairs	short	0.1442	Pipeline proven; eval accuracy 0.94
Run 1 (foundation corpus)	May 8	48,621 pairs (security QA + NIST SP 800-61/53)	2 h 53 m	0.2846 (eval 0.2727)	Baseline; corpus too generic
Run 2, first attempt	May 12	54,937 pairs, 20 scraped sources	failed at 4 h 23 m	—	SSH disconnect killed the run (no tmux)
Run 2 retry (run_7a20542c) → **v2**	May 13	same 54,937 pairs	14,293 s	0.1894 (eval 0.2024)	**Production v2**; adapter 104MB, merged 15GB
Run 3 (run_f94737d0) → v3	May 15	20,572 pairs, Azure/GCP/Vault expansion	~4 h	0.1987	Superseded by v4
Run 4 (run_9669d13e) → **v4**	May 16	27,597 pairs + cloud architecture patterns	17,098 s	**0.0057**	**Current production v4**

The corpus behind v2 came from 20 sources scraped into roughly 625 JSONL files totaling 15GB: cloud-provider documentation, Kubernetes runbooks, the Google SRE Book, HashiCorp documentation, CNCF project docs, engineering incident postmortems, tagged StackOverflow questions, and 17 engineering blogs. Deduplication used Jaccard similarity at a 0.92 threshold; an earlier 0.80 threshold incorrectly removed 67% of valid pairs because incident descriptions share boilerplate phrasing while differing in remediation, a miscalibration we corrected on April 24.

5.3 Evaluation and deployment gate

Evaluation compares each HelixModel iteration to a frontier-model baseline (Claude Sonnet 4.6 via AWS Bedrock) on 200 held-out cloud-operations incidents never seen in training, across five dimensions with explicit pass thresholds. The deployment gate requires at least 3 of 5 dimensions passing.

Table 6. *Five-dimension deployment gate on 200 held-out incidents. Thresholds are relative to the frontier baseline; lower is better for the last two rows. Per-dimension v2/v4 values from v1.0 are withdrawn pending a reproducible re-run (see Section 5.3).*

Dimension	Pass threshold	v2	v4
Incident routing accuracy	≥ 95% of baseline	Pending	**Pending**
Severity accuracy	≥ 95% of baseline	Pending	**Pending**
Remediation BERTScore F1	≥ 95% of baseline	Pending	**Pending**
Escalation false-positive rate	≤ 105% of baseline	Pending	**Pending**
Hallucination rate (invented service names)	≤ 105% of baseline	Pending	**Pending**

In-lab evaluation recorded v2 passing 3 of 5 dimensions (deploy) and v4 passing 5 of 5 (strong deploy), and those results drove the promotion decisions recorded at the time. The raw output files behind those runs resided on the lab instance and were not retained at teardown. Because our evidence standard requires a retained artifact for every published figure, the per-dimension percentages reported in v1.0 of this paper are withdrawn; Table 6 marks them pending. One retained harness artifact survives (April 14, 2026, against a pre-v2 checkpoint): it records routing at 99.0% of baseline and hallucination rate slightly better than baseline, and it also exposed a measurement fault in the severity dimension, which returned 0.0000 for both models and was incorrectly scored as a pass by the relative criterion. The harness defect is logged for repair, and a reproducible re-evaluation of the current production model is scheduled; its dated output will replace Table 6 in the next revision. v4 is deployed as a 20% weight component in the production multi-model consensus service alongside two larger models at 40% each, gated off by default until first customer onboarding.

5.4 Cost structure

Training cost per iteration was approximately $8.50, all of it Bedrock API spend for synthetic-data generation, because H100 time was covered by Innovation Lab credits. Measured inference economics: roughly $0.003 per frontier-model call versus roughly $0.0001 per HelixModel call amortized on H100, about 30×. On post-lab production hardware (T4-class GPU on EKS via vLLM with a 4.9GB Q4_K_M GGUF quantization), the gap narrows to roughly 10×, which still inverts the unit economics of high-volume structured inference.

6. Inference-Efficiency Techniques from Competition Constraints

In parallel with the benchmark, we entered the ARC Prize 2026 abstract-reasoning competition. The relevance is not the leaderboard; it is that the competition imposes the same constraints as our

production prediction cycle: a hard wall-clock budget (8.5 hours for 240 puzzles), no network access, fixed VRAM, and exact-match scoring. Techniques that survive those constraints transfer directly.

The solver evolved from a single 8B model with test-time-compute program synthesis (describe the transformation rule, generate a candidate program, verify against training pairs, retry up to 10 times) to a three-model sequential consensus: a 70B deep reasoner (5 passes), a 14B volume voter (15 passes), and an 8B domain specialist (10 passes), for 30 votes per puzzle under cross-model majority vote, with models loaded sequentially to fit a 53GB working set in available VRAM. Internal score estimates rose from 3 to 8% for the single-model baseline toward an estimated 25 to 35% for the full stack; we label these as internal estimates, not official leaderboard results.

Five techniques were back-ported to production:

- **Reasoning-trace distillation.** Replace raw outcome labels in fine-tuning data with full reasoned traces from a stronger model, so the student learns the reasoning path rather than the answer alone.
- **Self-correction loop.** On schema-validation failure, show the model its exact error and the expected structure and let it retry; this recovers roughly 30% of JSON failures in our pipeline.
- **Skill injection.** Inject a small library of pre-tested primitives into the prompt so the model composes verified building blocks instead of writing raw logic; fewer syntax errors per attempt.
- **Adaptive test-time compute with early exit.** Run a cheap fast phase first and exit on high confidence; escalate to a fuller, slower phase only when needed.
- **Aggressive quantization for serving.** LoRA merge to f16, then Q4_K_M GGUF (4.9GB), enabling deployment on inexpensive single-GPU instances at a measured 29% serving-cost reduction.

The adaptive early-exit pattern generalizes into our quantum-classical hybrid: a fast classical phase (confidence fusion, exit at confidence ≥ 0.85), a full classical phase (three-lane LLM inference, exit when lanes agree), and a quantum phase (annealing or kernel methods) reserved for the genuinely hard combinatorial residue where Section 4 says advantage is plausible.

7. Engineering Failures and What They Cost

Compute-constrained teams should plan for the failure modes below; each cost us real time on borrowed hardware.

- **SSH disconnect killed a training run (May 12).** 4 hours 23 minutes of a fine-tune lost at 80% of epoch 2 because the process was attached to the SSH TTY. Every subsequent multi-hour job ran inside tmux. This is the cheapest lesson in the paper and the most common failure we hear from other teams.
- **Disk exhaustion mid-merge (May 15).** Three retained full checkpoints plus a 20GB backup directory filled the 500GB NVMe during a model merge. The fix is procedural: delete checkpoint N−2 when checkpoint N completes, as a step in the training script rather than a habit.

- **Dependency API break (May 8).** A major-version bump of the training library renamed a constructor argument and removed another, costing 2 hours of debugging. Pin exact dependency versions for anything that touches a multi-hour job.
- **Vendor removed from cloud marketplace (April 15).** D-Wave devices left AWS Braket mid-program. The SDK migration itself was a small change, but it propagated through documentation and configuration in eight places. Cloud marketplace listings are not permanent; keep a direct-SDK fallback for any hardware dependency.
- **Deduplication threshold miscalibration (April 24).** A Jaccard threshold of 0.80 removed 67% of valid training pairs because operational text shares boilerplate. Deduplication metrics must match the task; for structured QA, similarity on the answer text is more meaningful than on the question text.
- **Untracked model versions (April 21).** Multiple runs pushed to the same model-repository tag with no mapping from deployed weights to training run. We now inject the git SHA at container build time and record it with every prediction, making version tracking impossible to skip.

8. Limitations

The claims in this paper are bounded by the following, in roughly descending order of importance.

- **Synthetic benchmark data.** Lane 1 uses 10,000 synthetic telemetry samples and Lane 2 uses 50 synthetic attack graphs. Both were constructed to be representative (chronological splits, MITRE-derived novelty definitions), but no claim here has been validated on customer-production data; that validation is the purpose of the beta program in Section 9.
- **Single-run comparisons without variance estimates.** Each lane comparison is one benchmark run. We report no confidence intervals, no repeated trials, and no statistical significance tests, so small margins (Lane 1's 3.8% precision gap, Lane 2's novelty gain) should be read as point estimates. Repeated-trial benchmarking is planned before any peer-review submission.
- **Lane 2 results are simulator-only.** The QAOA gains exist on a noise-free state-vector simulator. Our own fidelity analysis (Section 4.3) implies current gate-based hardware would destroy the result. We report it as a conditional, hardware-gated finding, not a deployable capability.
- **Lane 3 baseline is greedy, not state-of-the-art.** A 23% improvement over greedy local search does not establish superiority over strong classical heuristics such as simulated annealing on CPU or mixed-integer programming. Benchmarking annealing against those baselines is required before the advantage claim can be considered settled.
- **Cost and ROI figures are modeled.** The $2.3M-per-year and 1,000-customer figures are linear projections from per-instance benchmark savings, not realized customer outcomes. Inference-cost comparisons amortize GPU cost under our utilization assumptions.
- **Evaluation scale.** The HelixModel deployment gate uses 200 held-out incidents, sufficient for an internal gate but small for a generalization claim. The competition score figures are internal estimates, not official leaderboard results.

- **Raw benchmark outputs were not fully retained.** The committed per-lane comparison reports, with their version-control history, constitute the primary audit record; raw benchmark JSON from the lab instance was not retained after program teardown. For fine-tuning, the retained artifacts are the machine-generated run summaries in the repository and the dated adapter pushes to our private model registry; per-checkpoint trainer state from the lab instance was not retained. Post-lab, artifact sync to durable storage is a mandatory final step of every run.

9. Conclusion and Future Work

Across 52 days we delivered three working systems and one measured answer. The answer: quantum methods won where the problem is combinatorial and maps natively to the hardware (discrete cost optimization on an annealer, +23% over greedy), showed simulator-only promise where the problem is combinatorial but the hardware is not ready (attack-path discovery via QAOA, +62% novelty at roughly 22% projected circuit fidelity on real devices), and lost cleanly where the problem is low-dimensional with piecewise-linear structure (anomaly detection, where XGBoost is better and 37× faster). The negative result shaped the product as much as the positive ones: Lane 1 runs XGBoost in production, Lane 2 runs a classical-first hybrid, and Lane 3 is the one place a QPU sits in the serving path.

The same cluster made small-model fine-tuning economically trivial: about $8.50 of API spend and four hours of H100 time per iteration produced an 8B model that cleared the five-dimension deployment gate against a frontier baseline in in-lab evaluation (re-verification scheduled; Section 5.3), at an order-of-magnitude lower serving cost.

Near-term work, in order: production deployment of the prediction stack on EKS with the quantized model on commodity GPU instances (June 2026); platform integration for service auth, customer sync, and prediction ingestion (July); a live annealing beta against five customers with a pre-registered success criterion of at least 20% savings over greedy on real workloads (August); and a fifth fine-tuning iteration on a 200,000-pair certification curriculum corpus, alongside the repeated-trial and stronger-baseline benchmarking that Sections 4 and 8 call for. A USPTO provisional patent application (No. 63/975,794, February 2026) covers elements of the consensus and prediction architecture; the baseline comparisons in this paper form part of its supporting record.

Acknowledgments

We thank the NVIDIA GCP Innovation Lab for H100 access, AWS for Braket simulator credits, D-Wave for Leap API access, and NVIDIA Brev for DGX Cloud provisioning. The fine-tuning corpus rests on the open documentation of the cloud-infrastructure community, including 94 public documentation sources, and on the Hugging Face trl and PEFT frameworks.

References

References

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. Proceedings of KDD 2016, 785–794. DOI: 10.1145/2939672.2939785

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. arXiv:2305.14314

D-Wave Systems (2023). D-Wave Advantage performance update. Technical report.

Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm. arXiv:1411.4028

Glover, F., Kochenberger, G., & Du, Y. (2018). Quantum bridge analytics I: A tutorial on formulating and using QUBO models. 4OR 17(4), 335–371.

Hagberg, A., Schult, D., & Swart, P. (2008). Exploring network structure, dynamics, and function using NetworkX. Proceedings of SciPy 2008, 11–15.

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). Supervised learning with quantum-enhanced feature spaces. Nature 567, 209–212. DOI: 10.1038/s41586-019-0980-2

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). LoRA: Low-rank adaptation of large language models. arXiv:2106.09685

MITRE Corporation. MITRE ATT&CK knowledge base. https://attack.mitre.org/

National Institute of Standards and Technology. SP 800-61 Rev. 2: Computer security incident handling guide; SP 800-53 Rev. 5: Security and privacy controls.

Pearl, J. (1988). Probabilistic reasoning in intelligent systems: Networks of plausible inference. Morgan Kaufmann.

Beijing Academy of Artificial Intelligence (2024). BGE-M3: Multi-lingual, multi-functionality, multi-granularity text embeddings. https://huggingface.co/BAAI/bge-m3

Foundation AI. Foundation-Sec-8B-Instruct model card. https://huggingface.co/fdtn-ai/Foundation-Sec-8B-Instruct

Cybersecurity and Infrastructure Security Agency. Known Exploited Vulnerabilities catalog. https://www.cisa.gov/known-exploited-vulnerabilities-catalog