FluxQuapacitors vs LABS: A CUDA-Q Acceleration Story

TEAM MEMBER DETAILS

Project Lead (Architect): Vedika Saravanan

GPU Acceleration PIC (Builder): Lim Wee Keat and Gabriel Ortega

Quality Assurance PIC (Verifier): Gabriel Ortega and Yash Singh

Technical Marketing PIC (Storyteller): Travis Martin

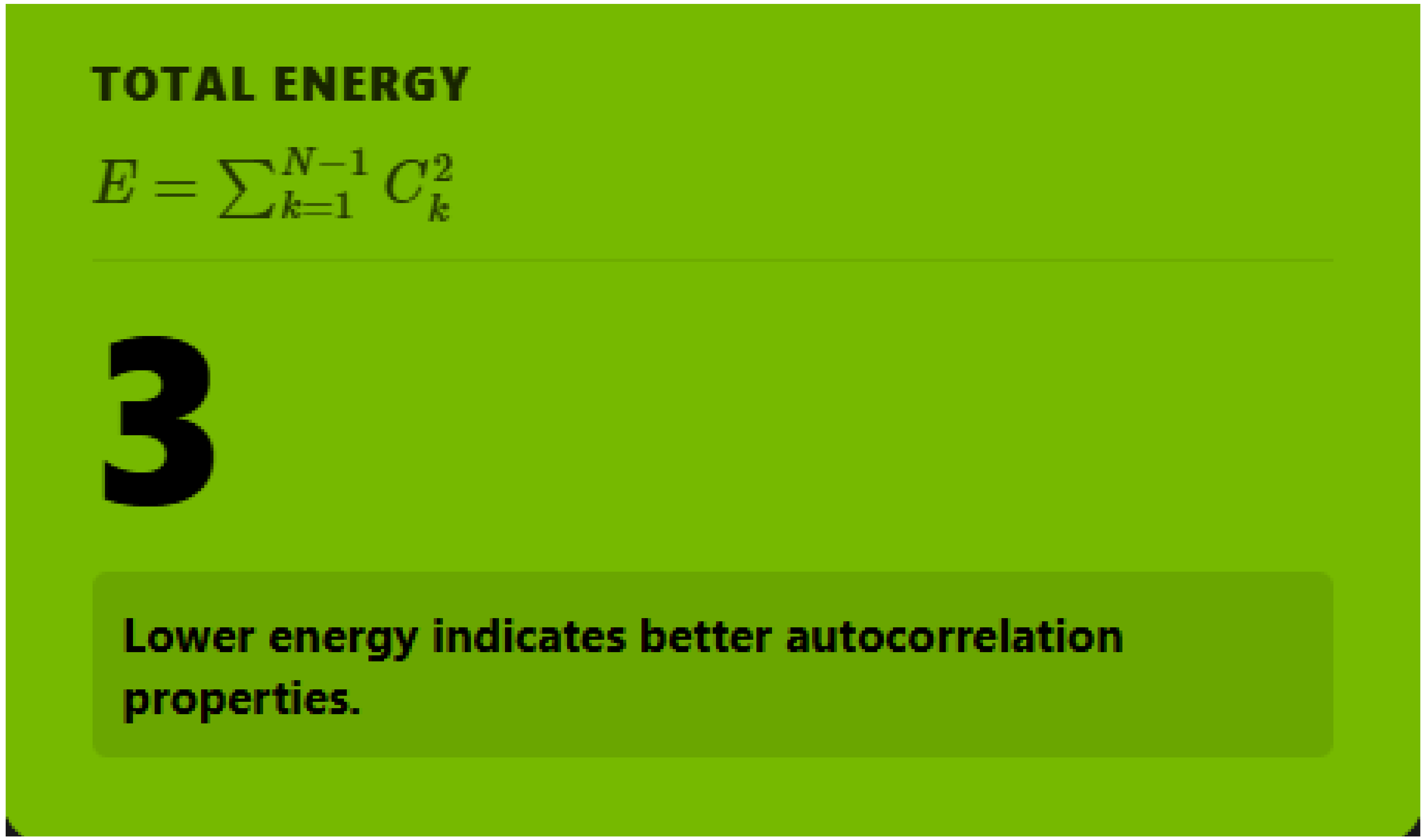
The LABS Problem

Low Autocorrelation Binary Sequence (LABS) is a fundamental problem in combinatorial optimization and signal processing.

- **Goal:** Find a binary sequence of length N that minimize aperiodic autocorrelation side-lobes → maximize merit factor / minimize energy
- **Scale:** Exponential search space of $\mathbf{2^n}$ configurations
- **Difficulty:** Highly rugged, non-convex energy landscape → classical heuristics often get trapped in local minima
- **Why it matters:** High-precision radar, deep-space pulse compression, and cryptographic keys

➡ LABS is therefore an ideal candidate for hybrid quantum-classical acceleration with CUDA-Q.

LABS as an Energy Minimization Problem



TOTAL ENERGY

$E = \sum_{k=1}^{N-1} C_k^2$

3

Lower energy indicates better autocorrelation properties.

Minimum Energy (Example, N = 7)

This energy value defines sequence quality — reaching the global minimum directly enables high-precision signal detection in hardware.

Plan and Pivot

Original PRD strat:

Objective: Scale and refine the Quantum-Enhanced Memetic Tabu Search (QE-MTS) for larger LABS instances.

- **QAOA optimization**
  - Replace fixed angles with adaptive optimization
  - Use COBYLA / SPSA to tune variational parameters (γ, β)
  - Improves quality of the initial candidate pool
- **GPU-accelerated simulation**
  - Migrate backend to NVIDIA cuQuantum SDK
  - Enables deeper circuits and larger qubit counts
- **Enhanced classical search**
  - Add adaptive tabu tenure and diversification
  - Prevents premature convergence in rugged, high-N landscapes

➡ Goal: Robust performance as LABS scales to larger N

What changed?

QAOA parameter strategy

- PRD: Fully variational QAOA loop using COBYLA / SPSA
- Final: Fixed ansatz parameters
- Why: Prioritized rapid sampling and clean hardware benchmarking isolation

Tabu Search configuration

- PRD: Adaptive tabu tenure and diversification
- Final: Fixed-tenure tabu search
- Why: Reduced classical overhead within the hackathon timeframe

CUDA-Q kernel constraints

- Unplanned change: Manual decomposition of R(ZZ)
- Implementation: CNOT → RZ → CNOT
- Reason: Ensure compatibility with the available CUDA-Q gate set

➡ Outcome: Faster iteration, stable benchmarking, and a fully CUDA-Q–compatible pipeline

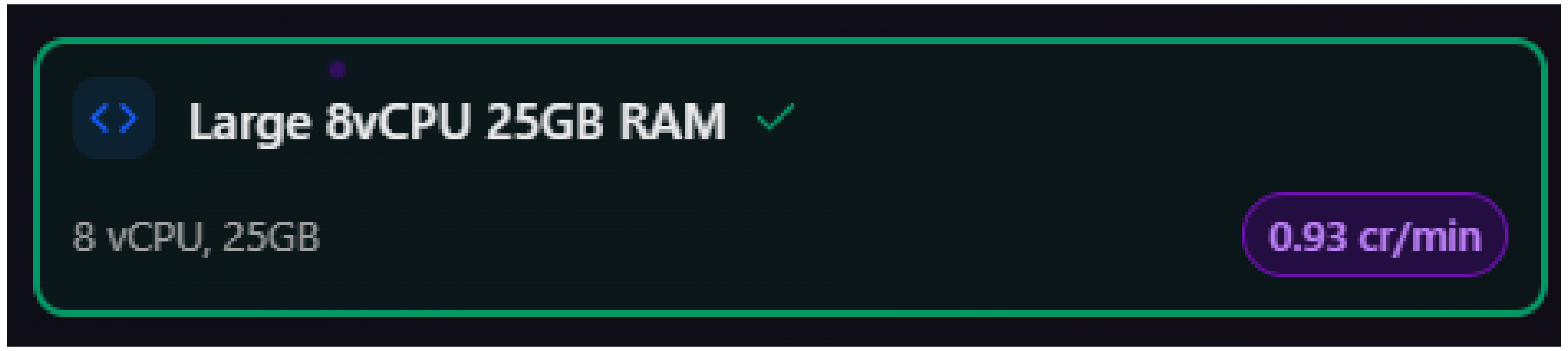
Results!!!

CPU vs GPU Benchmarking Setup

Goal: Compare solution quality and runtime across CPU and GPU backends under identical conditions.

Hardware compared

- CPU: qBraid Large 8 vCPU (25 GB RAM)
- GPUs: NVIDIA Brev GPUs
- Control: Local RTX 4080 (physical GPU)



Large 8vCPU 25GB RAM ✓

8 vCPU, 25GB

0.93 cr/min

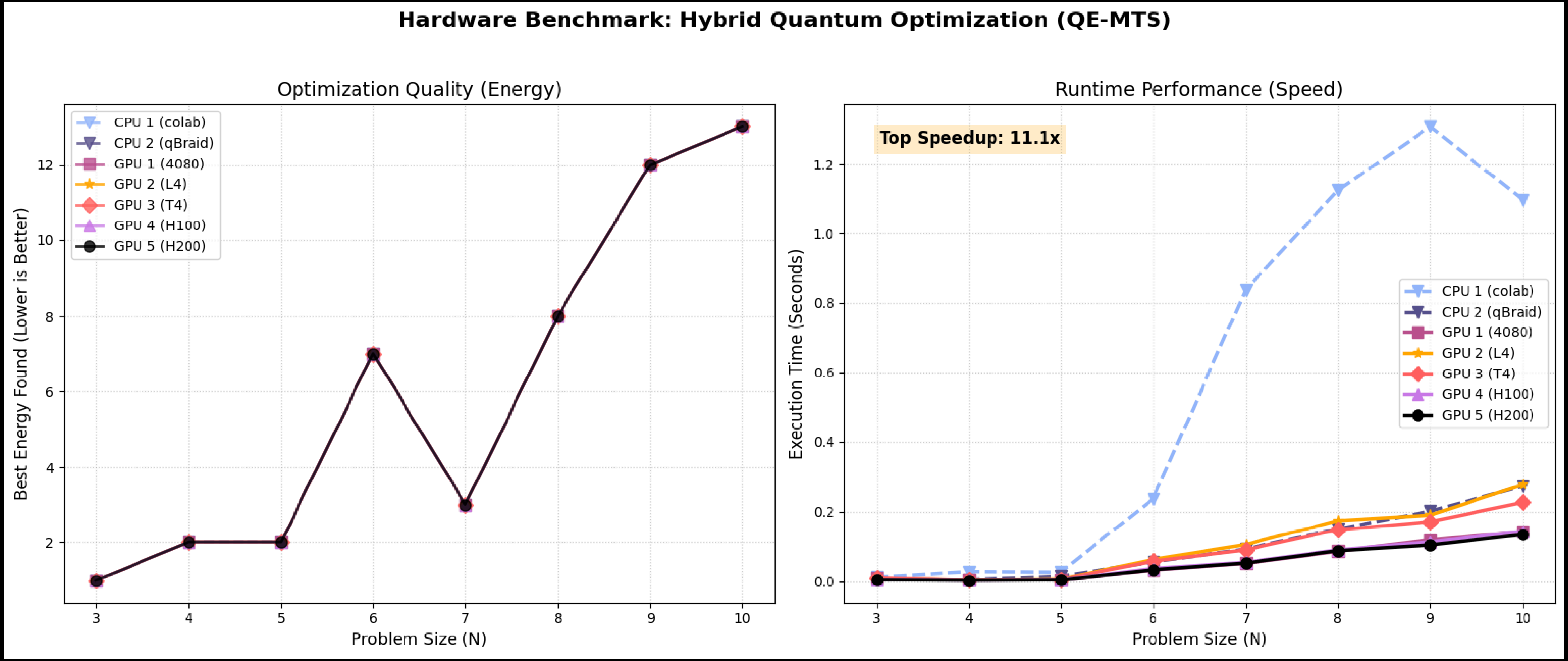
Experimental controls

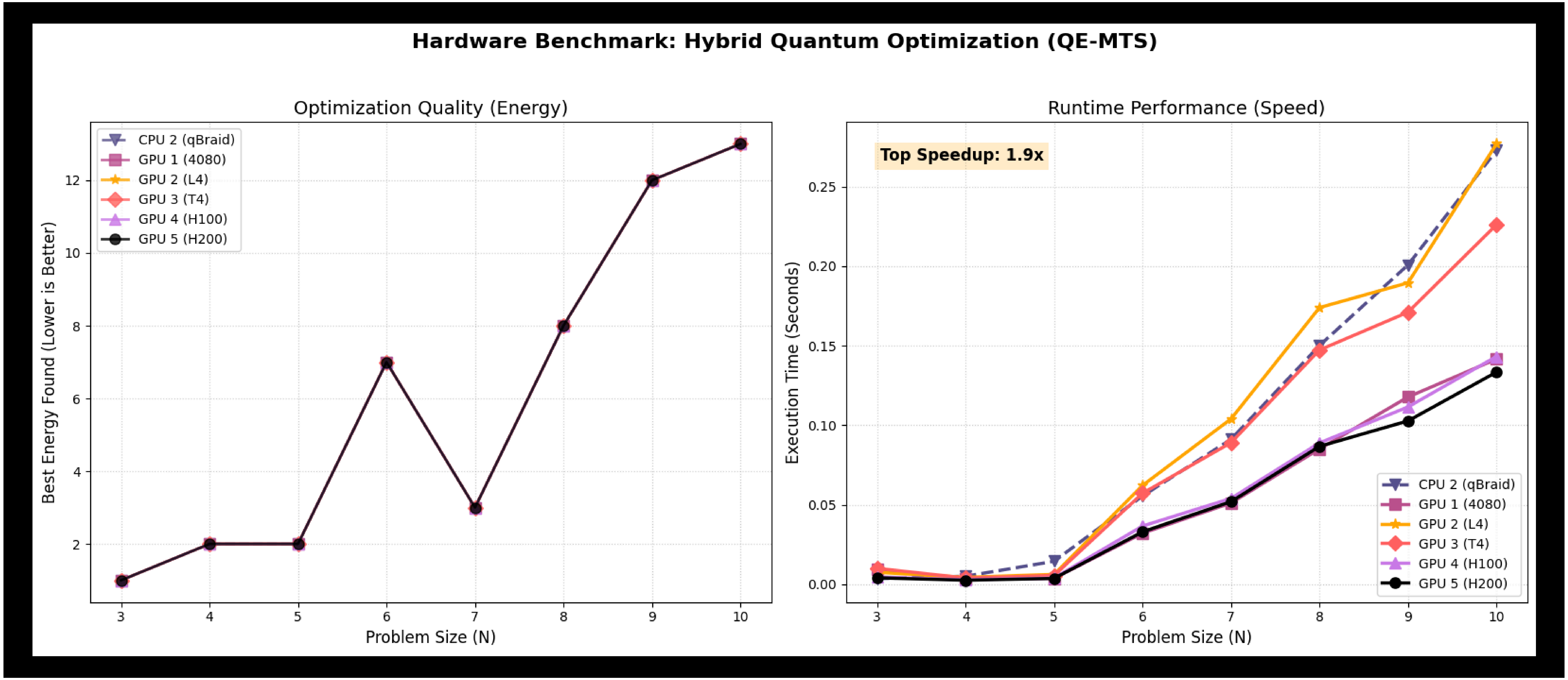
- Same algorithm, parameters, and problem instances
- Solution graph held constant
- Only variable: hardware backend

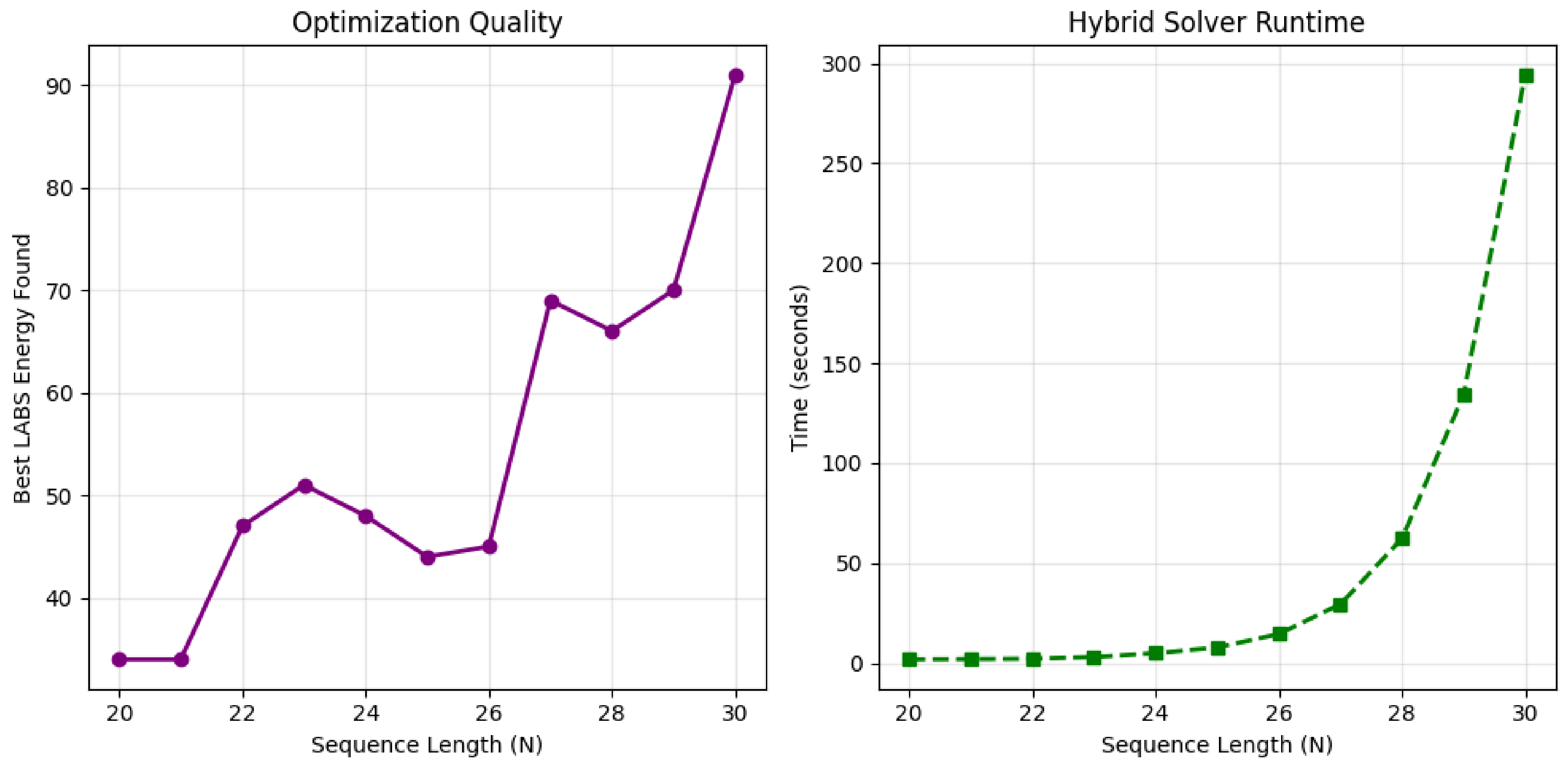
Why this matters

- Isolates true hardware effects
- Ensures fair, apples-to-apples comparison
- No algorithmic shortcuts or hidden optimizations

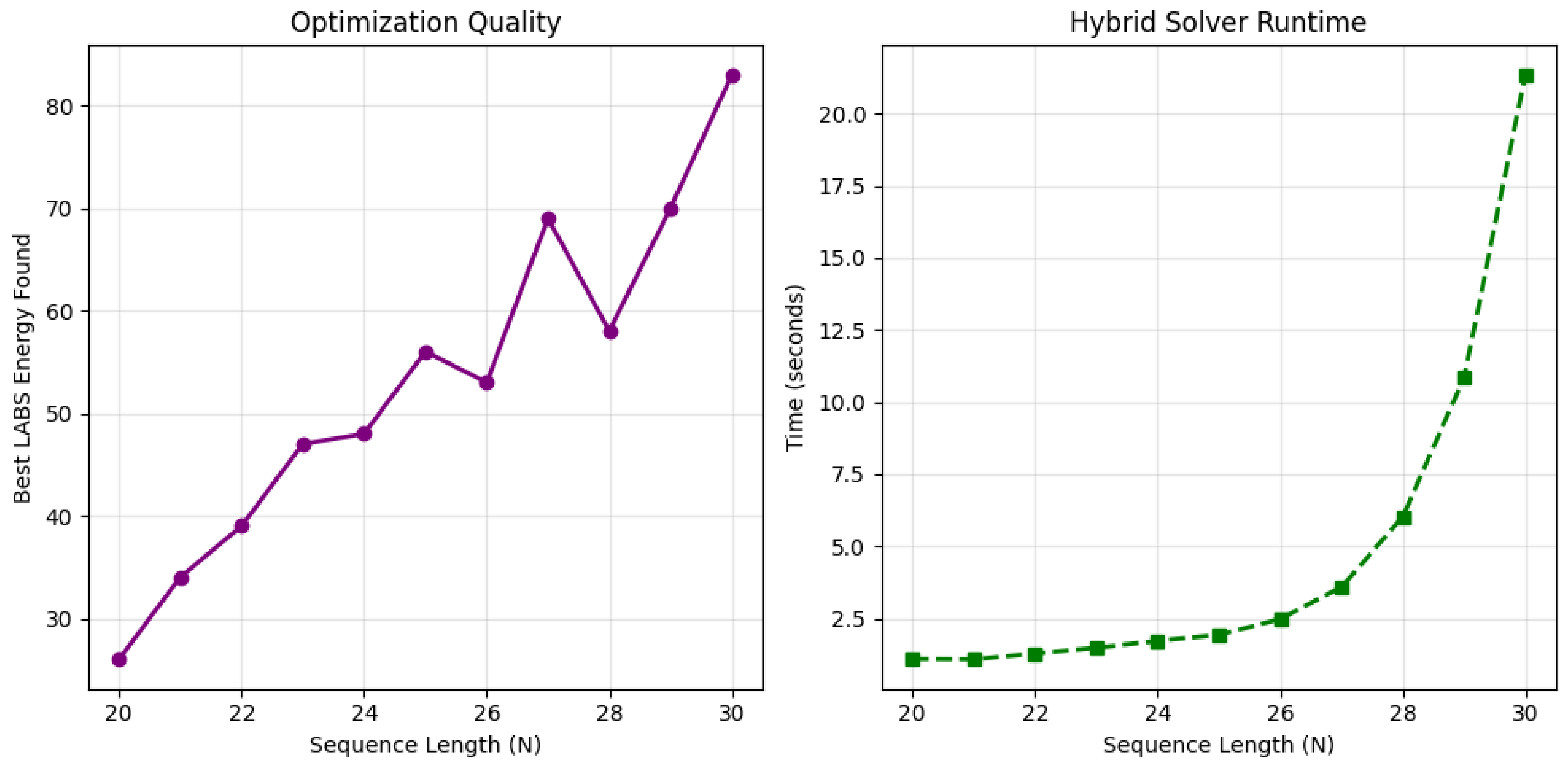
➡ Outcome: Differences in performance reflect hardware capabilities only


Hardware Benchmark: Hybrid Quantum Optimization (QE-MTS)
Optimization Quality (Energy)
Best Energy Found (Lower is Better)
Problem Size (N)
Runtime Performance (Speed)
Execution Time (Seconds)
Top Speedup: 11.1x
CPU 1 (colab)
CPU 2 (qBraid)
GPU 1 (4080)
GPU 2 (L4)
GPU 3 (T4)
GPU 4 (H100)
GPU 5 (H200)

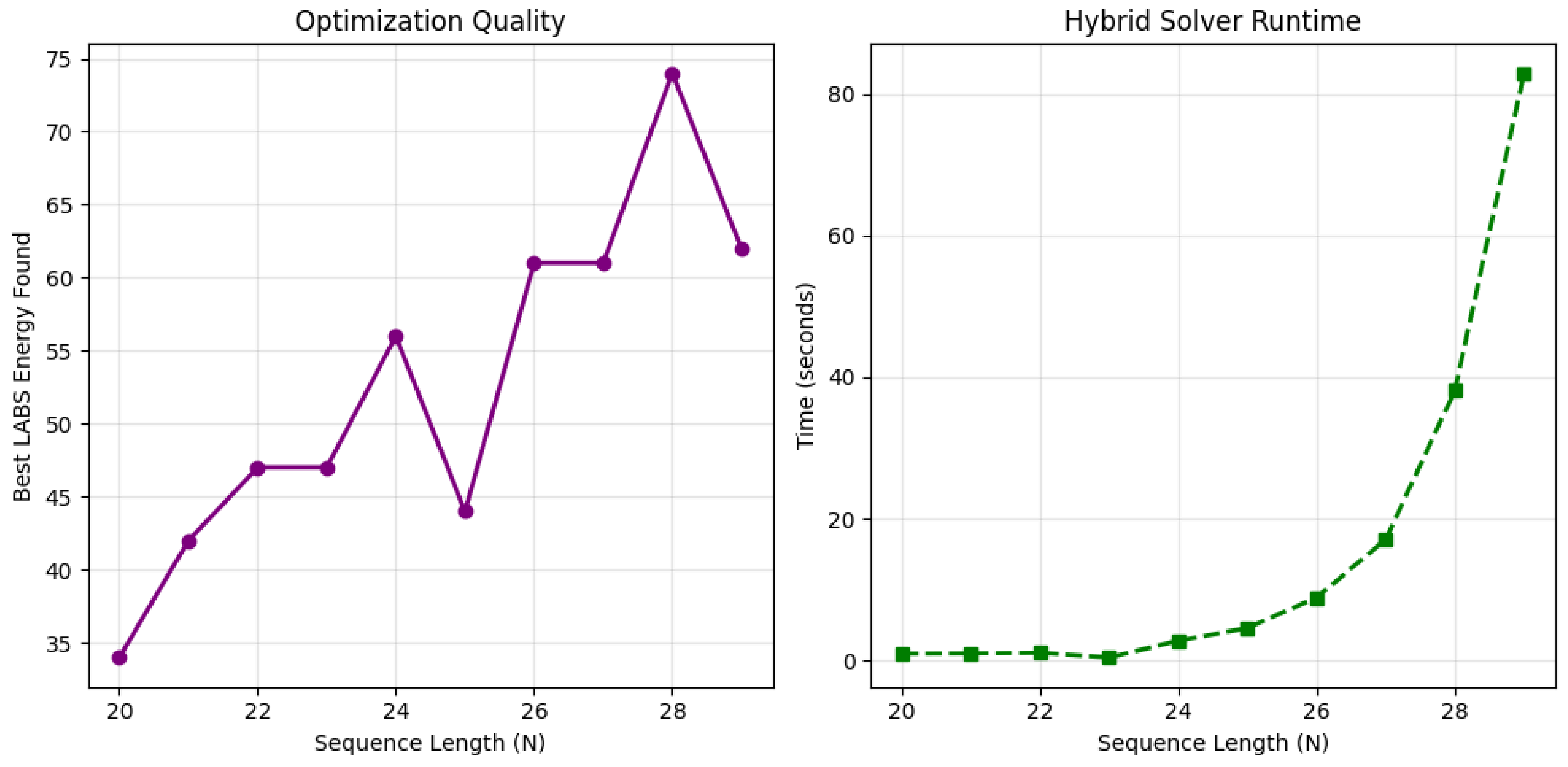

Hardware Benchmark: Hybrid Quantum Optimization (QE-MTS)
Optimization Quality (Energy)
Best Energy Found (Lower is Better)
Problem Size (N)
CPU 2 (qBraid)
GPU 1 (4080)
GPU 2 (L4)
GPU 3 (T4)
GPU 4 (H100)
GPU 5 (H200)
Runtime Performance (Speed)
Top Speedup: 1.9x
Execution Time (Seconds)
Problem Size (N)



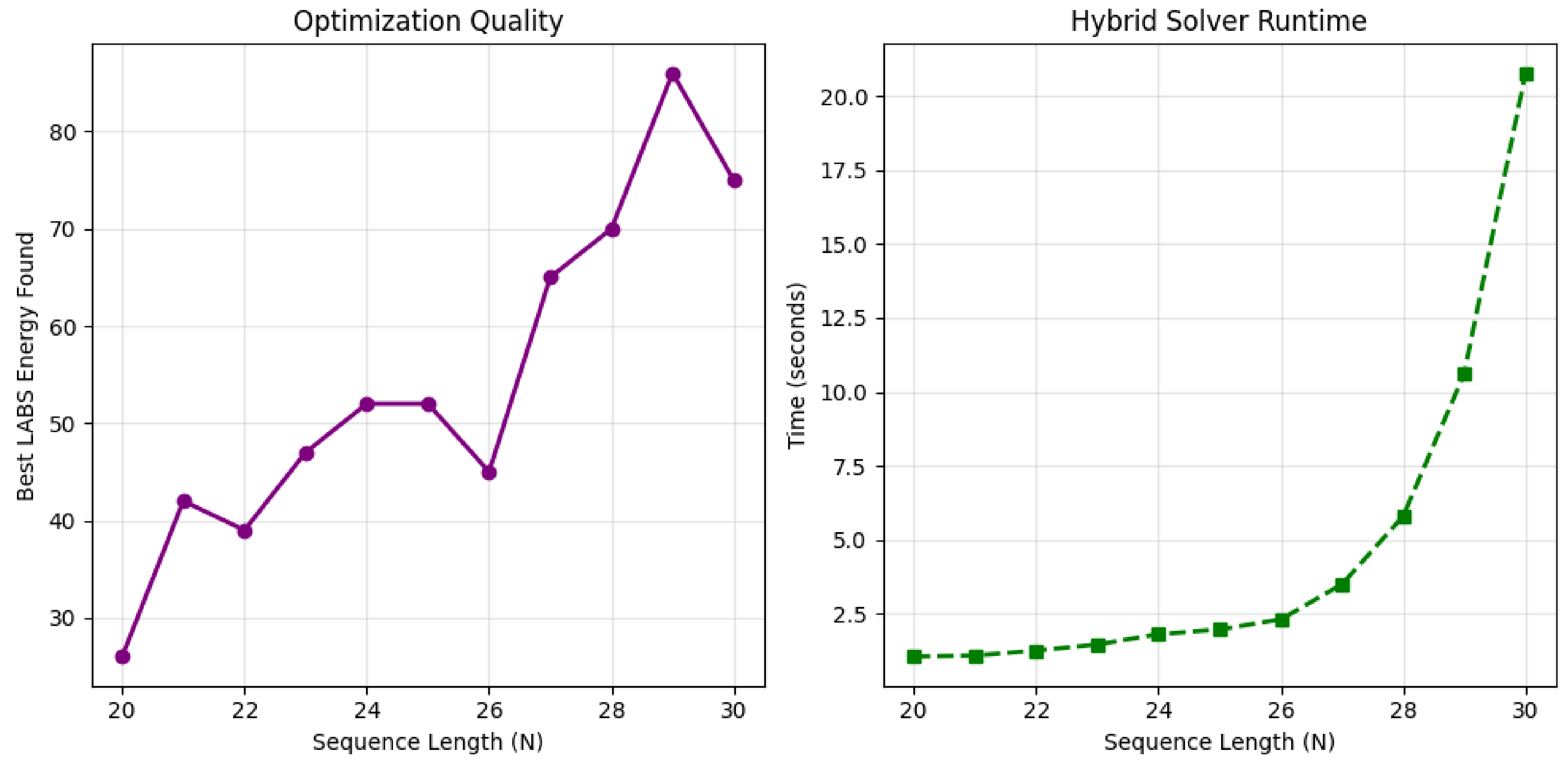
L4



H100



RTX 4080



H200

Summary

Key Observations & Takeaways

- Algorithm quality alone does not determine performance at scale
- Hardware characteristics become dominant as problem size increases
- **Small scale (n = 3 → 10):**
  - No meaningful difference between CPU and GPU results
  - Output is stable across hardware types
- **Larger scale (n = 20 → 30):**
  - Significant variation in outputs for the same algorithm
  - Differences driven primarily by GPU quality and architecture
- **Runtime performance is not a reliable predictor of solution quality**
- **Example:**
  - RTX 4080 vs H200 show similar runtimes
  - Yet produce different energy outputs
- **Implication:**
  - GPU choice impacts numerical outcomes, not just speed
  - Hardware-aware evaluation is critical for large-scale workloads