Quantum versus Classical Benchmarking

Design reproducible comparisons that separate correctness, uncertainty and performance.

Core 21 min lesson Dr. Charith Jayasekara

By the end, you'll be able to

- Freeze benchmark tasks and accounting rules.
- Use credible tuned classical baselines.
- Report variability, setup and repeated-execution costs.

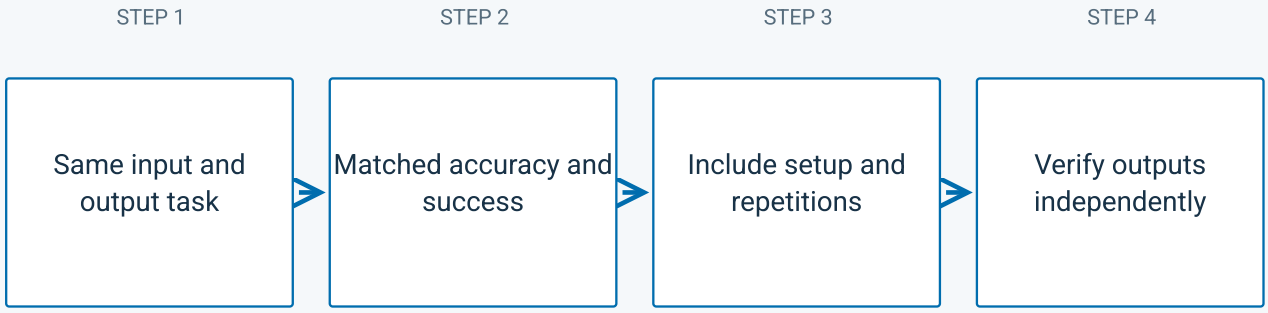
A benchmark should answer a reproducible question

A quantum-versus-classical benchmark compares specified implementations on a defined task. It should help a reader understand what was measured, which assumptions were shared and what conclusion the evidence supports. A headline ratio without those details can conceal different outputs, accuracy targets or amounts of preprocessing.

Prerequisites: simulation evidence, complexity and at least one algorithm. Primary benchmarking methodology explains why speedup definitions, instance selection and scaling choices matter. Rønnow and colleagues.

WORKED DIAGRAM · A39

A reproducible comparison



A tiny Grover circuit demonstrates a query mechanism; timing its classical statevector simulation does not measure quantum-processor speedup.

Freeze the task before the comparison

Define input instances, output format, acceptable error and target success probability. State the resource metric and accounting boundaries. If one method outputs a quantum state while another prints a full vector, they have not solved

the same output task. If one method is allowed a costly precomputed data structure, explain whether the other receives comparable preprocessing.

For an optimisation benchmark, define quality and feasibility. For sampling, define the distributional approximation requirement. For a simulation, define the observable and uncertainty. The benchmark's main question should be understandable before any performance plot is shown.

Worked example: a small search benchmark

Take one marked item among four, with coherent oracle access. One ideal Grover iteration returns the marked label with certainty. A classical checker querying candidates until it finds the marked one has an expected 2.5 checks for a uniformly hidden label if it also checks the final candidate rather than inferring it from the promise. State that convention; with inference, the final-query accounting changes.

This comparison demonstrates a query-model mechanism on a tiny instance. Timing a classical statevector simulation of Grover against a simple classical loop measures the simulator implementation, not a quantum processor's query cost. The simulator stores and updates all amplitudes classically, so its overhead belongs to a different execution model.

Suppose an actual coherent oracle is expensive to implement. Then the query count must be combined with its gate cost, preparation, error handling and measurement. A lower query count can coexist with a larger physical runtime on a small task. Both statements can be true because they count different resources.

Make the final-query convention visible in the classical reference value.

$$\mathbb{E}[Q_{\text{check all}}] = \frac{1+2+3+4}{4} = 2.5, \qquad \mathbb{E}[Q_{\text{infer last}}] = \frac{1+2+3+3}{4} = 2.25.$$

The hidden marked label is uniform among four candidates. The first procedure verifies the final candidate; the second infers it from the unique-mark promise after three failures. Compare procedures using the same output and verification requirement.

Choose credible classical baselines

Use algorithms that exploit the problem's known structure. A sparse linear system should not be compared only with dense elimination if an iterative sparse solver answers the same question. A graph optimiser should include strong problem-aware methods. A circuit-sampling comparison should consider specialised simulators where applicable.

Document versions, hardware, parallel resources, precision and tuning. A fair study need not use identical hardware technologies, but it must explain the chosen cost metric. Energy, elapsed time, monetary cost and processor count can produce different rankings. Do not hide a large classical cluster behind one number while treating every quantum control component as irrelevant, or vice versa.

Include the complete quantum workflow

Count model construction, state preparation, compilation, parameter training, repeated circuits, readout, mitigation and classical postprocessing where they belong to the task. Distinguish one-time setup from per-instance and per-query cost. If setup is amortised across many uses, show the workload that justifies it.

For variational methods, include the optimisation loop rather than only the final circuit. For estimation, include the shots needed for the reported precision. For fault-tolerant proposals, include code and resource-state assumptions. A concise resource table can make those boundaries clearer than an unexplained aggregate runtime.

Use statistics that match the experiment

Repeat stochastic runs and use enough independent instances to estimate variability. Show success distributions, quality curves or confidence intervals. A best-of-many result is not comparable with an average-of-one result. If a method times out, apply a declared rule rather than silently deleting the hard instance.

For time-to-success calculations, state the independence model and uncertainty in the estimated success probability. Rare success is difficult to estimate from few trials. A zero-success sample does not prove the probability is exactly zero, and a single success does not establish a stable high rate. Statistical evidence should match the strength of the conclusion.

Separate correctness from performance

Validate outputs independently before comparing speed. Small instances with known solutions catch implementation errors. Cross-check conventions such as bit ordering and Fourier signs. A fast incorrect answer is not an algorithmic improvement. Conversely, a correct small prototype does not establish scaling at sizes that were never tested.

For hard sampling tasks, verification may be limited. State what a statistic tests and which alternative distributions it might accept. Primary sampling-complexity research studies both hardness and verification questions, preserving their assumptions. Bouland and colleagues. A benchmark should not imply full distribution certification from one convenient score without justification.

A reproducible study plan

Predefine the task and instance-generation method. Establish correctness references. Allocate tuning budgets. Freeze selected configurations. Run independent evaluation trials under the declared accounting rule. Publish sufficient instance data, parameters and result summaries to reproduce the comparison. Then state a conclusion whose scope matches the evidence.

For a student project, compare exact probability predictions, sampled frequencies and runtime separately. This prevents a single plot from conflating mathematical correctness, statistical accuracy and implementation speed. Those are three useful results, but they require different interpretations.

What a benchmark can conclude

An experiment may show that one method is faster on the tested cases under its accounting rule. It may reveal a crossover, a scaling trend or a bottleneck. It generally cannot prove that no better classical algorithm exists. Preserve that distinction without dismissing the practical value of the observed result.

Benchmarking is an ongoing scientific process. New classical methods, compilation improvements and hardware changes can revise comparisons. A well-documented result remains useful because others can locate exactly which assumption or implementation improved, rather than arguing over a vague claim about “quantum speed.”

Exercises and worked answers

Exercise 1. A quantum method reports only final execution time after ten thousand training evaluations. What comparison information is missing?

Check the worked answer

The training cost, selection policy, total shots and whether that cost is amortised across a stated workload. Include an end-to-end result or clearly label the narrower inference-only metric.

Exercise 2. Why should benchmark instances used to select parameters be separated from final evaluation instances?

Check the worked answer

Otherwise selection can exploit accidental features of those cases and exaggerate general performance. Held-out evaluation tests the frozen method under the declared distribution, just as in ordinary empirical algorithm and machine-learning studies.

Check your understanding

PAUSE & THINK

Does a slow browser Grover simulation refute Grover's query theorem?

- [ ] Yes
- [ ] No; it measures a different execution model
- [ ] Only when the register has four states

Answer: No; it measures a different execution model. A statevector simulator is a classical implementation with its own overhead. The theorem counts coherent oracle queries in a quantum computation model.

Sources

- Rønnow and colleagues
- Bouland and colleagues