The Quantum Tortoise and the Classical Hare: When Will Quantum Computers Outpace Classical Ones and When Will They Be Left Behind?

Sukwoong Choi, William S. Moses, Neil Thompson

III. Hardware Differences

A. Trends in the number of qubits in quantum computers

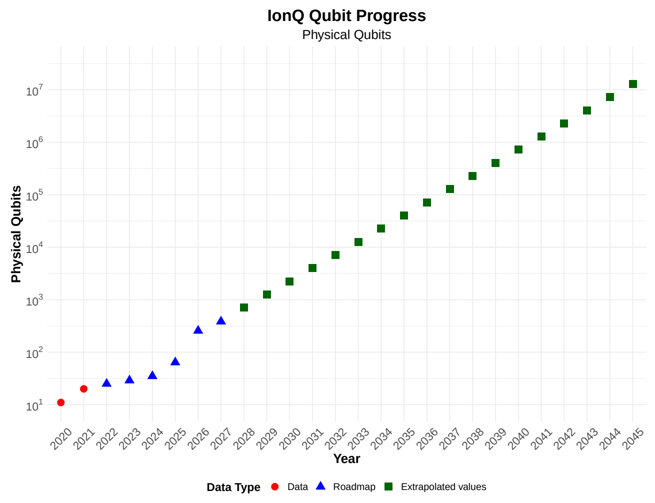

Fig. S1: **IonQ's Projected Physical Qubit Growth:** Plot showing the number of physical qubits over time, with real data (Red), roadmap (Blue), and extrapolated values (Green) based on an exponential growth model.

IonQ's trapped-ion quantum computing approach highlights a unique trade-off between high qubit connectivity and slower gate speeds. Unlike superconducting qubits, which often rely on nearest-neighbor connectivity, trapped-ion systems allow any qubit within the same trap to interact directly with any other qubit. This high connectivity simplifies the implementation of quantum circuits, particularly for algorithms requiring extensive qubit interactions or complex entanglement, but comes at the cost of slower gate speeds. For instance, a study demonstrated entangling gate operations with durations as short as 15 microseconds (15 μs) [1]. While this is a significant achievement, it is still slower than the gate speeds typically achieved in superconducting systems.

Extrapolating from IonQ's roadmap, as detailed in Figure S1, suggests that exponential growth in the number of physical qubits could potentially lead to approximately 10^3 qubits by 2030 and 10^6 by 2040. Achieving these milestones, however, will require overcoming significant technical challenges, including improving gate speeds, minimizing error rates, and maintaining coherence among qubits in larger systems. These challenges highlight the broader difficulties of scaling quantum computing technologies, regardless of architecture.

C. The speed difference between classical & quantum computers

Table S1 expands on the specifications discussed in Section C of the main article. It estimates the quantum performance gap by comparing the cost per serial and parallel operations, offering a comprehensive view of the differences between classical and quantum systems.

	Classical	Quantum	Ratio
Speed (serial cost)	5GHz[1]	2MHz[2]	$10^3 - 10^4\times$
		+ error-correction overhead [100×]	$10^5 - 10^6\times$
Cost	10^{14} FLOPS/$[3]	10^8 gate ops/$[4] + error-correction overhead [100×]	$10^6\times$ $10^8\times$

TABLE S1: **Calculation of quantum performance gap, both from cost per serial operation and cost per (parallel) operation.**

IV. Algorithmic Differences

Table S2 draws on the analysis from [5] to list the computational complexities of classical algorithms, for use in interpreting Figure 5 for each problem.

	Classical Algorithm
Exp	3-Graph / 4-Graph Coloring (NP) Subset-Sum (NP) Vertex Cover (NP) Set-Covering (NP) Maximum Cut (NP) The Traveling-Salesman Problem (NP) Enumerating Maximal Cliques (NP) n-Queens (NP) Graph Edit Distance Computation (NP [6]) Turnpike NFA to DFA Conversion Dependency Inference BCNF Decomposition 4NF Decomposition Discovering Multivalued Dependencies Finding Frequent Item Sets Motif Search Minimum Wiener Connector Change-Making Median String Factoring
n^4	Longest Path on Interval Graphs Determinant using Integer Arithmetic
n^3	Grobner Bases Maximum-weight Matching MDPs for Optimal Policies Link Analysis (Indegree) Integer Relation
$n^{2.38}$	Parsing
$n^{2.188}$	Transitive Reduction
$n^2 \log n$	Nash Equilibria Entity Resolution 2-D Elliptic Partial 3-D Elliptic Partial All-pairs Shortest Path
n^2	Factorization of Polynomials over Finite Fields Cryptanalysis of Linear Feedback Shift Registers Translating Abstract Syntax Trees into Code Digraph Realization Sequence to Graph Alignment (Linear Gap Penalty) Rod Cutting Frequent Words
$n^2 / \log n$	Sequence Alignment
$n^2 / \log^2 n$	Longest Common Subsequence
$n^{1.5}$	Max Flow
$n^{1.188}$	Maximum Cardinality Matching
$n^{1.186}$	Matrix Multiplication
$n^{1.185}$	Linear Programming
$n \log n \log \log$	Greatest Common Divisor
$n \log n$	LU Decomposition Multiplication Discrete Fourier Transform (QEXP) Cyclopeptide Sequencing Comparison Sorting Line Segment Intersection Convex Hull Closest Pair Problem SDD Systems Solvers Polygon Clipping Nearest Neighbor Search Voronoi Diagrams Delaunay Triangulation Weighted Activity Selection
$n \log n$	Single-interval Scheduling Maximization (Unweighted)
$n \log n \log \log$	Greatest Common Divisor
$n \log \log n$	Shortest Path Duplicate Elimination

[1]Venten et al. (2022) [2]

[2]Calculated as gate operations per second from single and multi-qubit gate timings. On a Google machine single qubit gate takes 20ns, with a two qubit controlled Z gate taking 32ns [3]. On an IBM machine, gate time averages are 555ns [4].

[3]https://en.wikipedia.org/wiki/FLOPS

[4]https://www.ibm.com/quantum/access-plans

n	Line Clipping
	Integer Sorting
	kth Order Statistic
	Linear System of Equations
	Strongly Connected Components
	Minimum Spanning Tree
	String Search
	Joins
	Cycle Detection
	Generating Random Permutations
	Minimum value
	All Permutations
	Huffman Encoding
	Constructing Eulerian Trails in a Graph
	Line Drawing
	Topological Sorting
	DFA Minimization
	Lowest Common Ancestor
	De Novo Gene Assembly
	Disk Scheduling
	Lossy Compression
	Stable Marriage
	Maximum Subarray
	Constructing Suffix Trees
	Longest Palindrome Substring
	Matrix Factorization for Collaborative Filtering
	Point in Polygon
	Polynomial Interpolation
	Deadlock Avoidance
	Page Replacements
	Recovery
$\log n$	Mutual Exclusion
	Self-Balancing Trees Insertion
	Self-Balancing Trees Deletion
	Self-Balancing Trees Search

TABLE S2: **Classical Runtime of Algorithms by Input Size:** Classical runtime of various computer science problems, with n taken as the conventional input size for the problem.

V. Sensitivity testing

To get the answers in the previous section, we estimated several parameters (speed difference, error correction overhead, etc.) and evaluation metrics (cost per serial operation and cost per parallel operation). In this section we consider how much our conclusions change if we assume more optimistic (or pessimistic) estimates.

A. Optimistic version

Here we consider how the thresholds for quantum economic advantage change if we make more optimistic assumptions for the performance of quantum computers, resulting in performance gap of 10^4. For example, this might arise if:

1) error correction codes improve,
2) quantum hardware produces fewer errors,
3) the costs of building and operating quantum computers falls more rapidly than classical computers

This optimistic version is shown at the middle of Figure 5. The overall results are qualitatively similar to the base case, but certain problems (e.g. when classical is n^3 and quantum is n), become attractive for problem sizes of only a few hundreds, that could be quite business-relevant.

B. Pessimistic version

Here we consider what thresholds for quantum economic advantage that we would predict if we instead made more pessimistic assumptions[5] that lead to a quantum performance gap of 10^8. This might arise if:

1) error correction becomes harder as systems get larger,
2) classical computers improve their price-performance faster than classical computers,
3) the connectivity issues for quantum computers make each logical qubit require more effective classical qubits

This pessimistic version is shown at the bottom of Figure 5. The overall results are again qualitatively similar to the base case, but now require problem sizes in the hundreds of millions or billions, putting many everyday business applications out of reach.

VI. Limitations to the analysis

A. Contextualizing Problem Size

It is important to note that these problem sizes n vary on the context of the problem. As a result, a very large n may be common in some problems, or uncommon in others. For example, we describe the runtime of Shor's algorithm as polynomial in n, where n is used to denote the number of bits of the value being factored. Therefore, a value of $n = 20$, denotes factoring a number of size $2^{20} \approx 10^6$. Thus if the using the analysis above you derived a threshold problem size of $n^* = 20$ in bits, that would correspond to 2^{n^*} being the threshold problem size in terms of the value of the integer being factored.

Understanding this conversion may often be important to practitioners. As an example, while the $n^* = C^2 = 10^{12}$ of may appear too large for most use cases to search through a list of size n, Grover's algorithm may be used to search through an exponentially large "list" of possible solutions to a problem. For example, one could apply Grover's algorithm to perform an exhaustive search over all $n = 2^m$ settings of a satisfiability problem of m variables. Grover's quadratic speedup on such a search problem would then result in $2^{m/2}$ evaluations of the search function [7]. The threshold problem size above would remain correct at $n^* = 10^{12}$, that would correspond to a threshold number of variables of $m^* = \log_2 n^* \approx 40$.

B. Precision

When analyzing the very small minimum problem sizes like in an algorithmic improvement from exponential time to polynomial, our order-of-magnitude analysis of the expected

[5]This does not consider improvements in algorithms, that would lead to problems changing complexity classes, such as if quantum computers could be efficiently simulated.

problem size becomes less accurate. This is because the additional constant overheads within the algorithms themselves, including constructing the correct quantum gateset or classical function, becomes more crucial. For example, when analyzing Shor's algorithm under our order-of-magnitude framework with an overhead constant $C = 10^6$, we find that factoring numbers of only a few bits will achieve speedup on a quantum computer. To determine more than rough estimates in these regime, significant consideration must be made for the the quantum-gate overhead constants, as well as lower-order constant overheads for the classical algorithms.

C. Quantum RAM / Data Loading

Thus far in our analysis, we have implicitly assumed that computing time will be the bottleneck to any computational task and thus ignored the time for loading the data. But this assumption can be wrong if the time to load the data is significantly longer than the time needed for the calculation. In these cases, the bottleneck will be the loading time of the data. This problem can arise particularly for quantum computers because of how they represent data or need to convert classical data into a quantum format.

In classical computing, doing an operation on one bit of information requires loading (at most) one bit from memory. Therefore, the cost of loading data is (at most) proportional to the number of operations. And thus, data loading costs cannot change the asymptotic scaling of classical algorithms.[6] By contrast, in quantum computing, each operation acts on a register of qubits, whose superposition and entanglement might require exponentially many bits of classical information to construct. In this case, the loading operations needed to build up the superpositions could be much greater than the operations needed for a calculation. In particular, if the amount of information needed to construct the superposition grows as the problem size grows, then the scaling of loading data could be worse than that of the calculation itself.

"Quantum RAM (qRAM)" [8] aims to provide a way of storing quantum information for later use that does not require conversion between classical and quantum formats, and hence avoids making the load operations too expensive.

Thus, loading data on a quantum computer can itself be the limiting factor on the scaling of a quantum algorithm if the data usage computations scale less well than the calculations on that data. It can also be a bottleneck if the data is stored in classical format, in which case there may be a computational penalty needed to convert it into a quantum-machine readable format. In this case, the time complexity of loading or storing data could be significantly worse than linear.

In practice, the time needed for loading thus becomes a lower bound on how well the quantum algorithm can do. For example, as we discussed with the DNA database example, an algorithm with a time complexity of $O(\sqrt{n})$ but where data loading is linear $O(n)$ will be limited by data loading. As such, an apparent computational benefit that a quantum algorithm has, might be erased by this effect.

[6]Assuming efficient "random-access" implementations that don't thrash, etc.

While much of our discussion focuses on superconducting qubits and IBM quantum computers, it is important to acknowledge that other QPU modalities, such as ion-trapped qubits, also play a significant role in the landscape of quantum computing architectures. Ion-trapped qubits, for instance, have significantly lower gate speeds than superconducting qubits, yet they benefit from lower overhead in terms of physical-to-logical qubit ratios. Each architecture has unique trade-offs, affecting connectivity and scalability; for example, ion-trapped systems offer higher connectivity among qubits compared to the more limited connectivity seen in many superconducting systems. These architectural variations contribute to differences in computational efficiency and error correction requirements.

D. Qubit connectivity

In our analysis so far, we have assumed that, once the error-correction overhead is paid, physical qubits can perfectly represent logical qubits. This assumption is optimistic because, in many cases, real qubits have limited connectivity compared to fully connected theoretical models; however, connectivity depends heavily on the QPU architecture. For instance, superconducting qubits in IBM's QPUs typically feature nearest-neighbor connectivity, where each qubit can only directly interact with a limited set of adjacent qubits. This restricted connectivity can increase circuit complexity for algorithms requiring interactions between distant qubits. Trapped-ion qubits, on the other hand, generally provide connectivity to other qubits in the common trap. While we do not account for it here, if the implied connectivity penalty for different architectures are known, they can be incorporated into this framework as an increase in the time complexity of the quantum algorithm on that hardware. Sometimes this distinction can be mitigated by clever quantum compilers that select and re-route computations to ensure necessary qubits are adjacent [9], [10]. However, solving for the optimal qubit assignment has been shown to be NP-complete [11], with approximation algorithms, like dynamic programming, being practical solutions. Such approximations can result in orders-of-magnitude additional costs [11].

Error rates also impact algorithm performance. Superconducting qubits, such as those in IBM's QPUs, have average single-qubit error rates around 0.1% and two-qubit error rates near 1%, often requiring substantial error correction. Although trapped-ion qubits generally have lower single-qubit error rates, their slower gate speeds can limit their advantage. Together, these differences in connectivity and error rates influence the feasibility of certain quantum algorithms, especially those requiring high-fidelity entanglement or intensive qubit interactions.

E. Hardware Variability Across QPU Modalities

While much of our discussion focuses on superconducting qubits and IBM quantum computers, it is important to acknowledge that other QPU modalities, such as ion-trapped qubits, also play a significant role in the landscape of quantum computing architectures. Ion-trapped qubits have both advantages and disadvantages compared with superconducting qubits. They benefit from lower error correction overhead (i.e. lower physical-to-logical qubit ratios) and better connectivity between qubits, but they also have significantly lower gate speeds than superconducting qubits. Collectively, these differences mean that ion-trapped qubits will get quantum economic advantage at different times for different problem sizes than superconducting qubits (as explored in the online appendix). Other architectures can be similarly analyzed with the Tortoise-Hare framework, by substituting in their respective features.

Another important caution relates to the exponential growth in the number of physical qubits shown in many QPU hardware roadmaps. Ensuring coherence, error correction, and sufficient qubit connectivity presents challenges that may grow in complexity with qubit count. This may make the growth in effective logical qubits substantially slower than the growth in physical qubits. Thus, these projections by QPU providers should be interpreted with caution.

References

[1] S. A. Moses *et al.*, "Universal control of a single trapped ion qubit with infinite coherence," *arXiv preprint arXiv:2305.03450*, 2023.

[2] M. Velten, R. Schöne, T. Ilsche, and D. Hackenberg, "Memory Performance of AMD EPYC Rome and Intel Cascade Lake SP Server Processors," in *Proceedings of the 2022 ACM/SPEC on International Conference on Performance Engineering*, vol. 1, no. 1. New York, NY, USA: ACM, apr 2022, pp. 165–175. [Online]. Available: https://dl.acm.org/doi/10.1145/3489525.3511689

[3] Quantum AI, Google, "Quantum computer datasheet," 2021, last accessed: 2022-09-12. [Online]. Available: https://quantumai.google/hardware/datasheet/weber.pdf

[4] IBM Quantum Research Blog, "Eagle's quantum performance progress," March 2022, accessed: 2024-12-04. [Online]. Available: https://www.ibm.com/quantum/blog/eagle-quantum-processor-performance

[5] Y. Sherry and N. C. Thompson, "How fast do algorithms improve?[point of view]," *Proceedings of the IEEE*, vol. 109, no. 11, pp. 1768–1777, 2021.

[6] Z. Zeng, A. K. Tung, J. Wang, J. Feng, and L. Zhou, "Comparing stars: On approximating graph edit distance," *Proceedings of the VLDB Endowment*, vol. 2, no. 1, pp. 25–36, 2009.

[7] A. Montanaro, "Quantum algorithms: an overview," *npj Quantum Information*, vol. 2, no. 1, pp. 1–8, 2016.

[8] V. Giovannetti, S. Lloyd, and L. Maccone, "Quantum random access memory," *Physical review letters*, vol. 100, no. 16, p. 160501, 2008.

[9] M. G. Pozzi, S. J. Herbert, A. Sengupta, and R. D. Mullins, "Using reinforcement learning to perform qubit routing in quantum compilers," *ACM Transactions on Quantum Computing*, vol. 3, no. 2, may 2022. [Online]. Available: https://doi.org/10.1145/3520434

[10] A. Cowtan, S. Dilkes, R. Duncan, A. Krajenbrink, W. Simmons, and S. Sivarajah, "On the Qubit Routing Problem," in *14th Conference on the Theory of Quantum Computation, Communication and Cryptography (TQC 2019)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), W. van Dam and L. Mancinska, Eds., vol. 135. Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2019, pp. 5:1–5:32. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2019/10397

[11] M. Y. Siraichi, V. F. d. Santos, C. Collange, and F. M. Q. Pereira, "Qubit allocation," in *Proceedings of the 2018 International Symposium on Code Generation and Optimization*, ser. CGO 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 113–125. [Online]. Available: https://doi.org/10.1145/3168822