



DOI:10.1145/3571725

What are the promising applications to realize quantum advantage?

BY TORSTEN HOEFLER, THOMAS HÄNER, AND MATTHIAS TROYER

Disentangling Hype from Practicality: On Realistically Achieving Quantum Advantage

OPERATING ON FUNDAMENTALLY different principles than conventional computers, quantum computers promise to solve a variety of important problems that seemed forever intractable on classical computers. Leveraging the quantum foundations of nature, the time to solve certain problems on quantum computers grows more slowly with the size of the problem than on classical computers—this is called *quantum speedup*. Going beyond quantum supremacy,² which was the demonstration of a quantum computer outperforming a classical one for an artificial problem, an important question is finding meaningful applications (of academic or commercial interest) that can realistically be solved faster on a quantum computer than on a classical one. We call this a practical quantum advantage, or *quantum practicality* for short.

There is a maze of hard problems that have been suggested to profit from quantum acceleration: from cryptanalysis, chemistry and materials science, to optimization, big data, machine learning, database search, drug design and protein folding, fluid dynamics and weather prediction. But which of these applications realistically offer a potential quantum advantage in practice? For this, we cannot only rely on asymptotic speedups but must consider the constants involved. Being optimistic in our outlook for quantum computers, we identify clear guidelines for quantum practicality and use them to classify which of the many proposed applications for quantum computing show promise and which ones would require significant algorithmic improvements to become practical and relevant.

To establish reliable guidelines, or lower bounds for the required speedup of a quantum computer, we err on the side of being optimistic for quantum and overly pessimistic for classical computing. Despite our overly optimistic assumptions, our analysis shows a wide range of often-cited applications is unlikely to result in a practical quantum advantage without *significant* algorithmic improvements. We compare the performance of only a single classical chip fabricated like the one used in the NVIDIA A100 GPU that fits around 54 billion transistors¹⁵

» key insights

- Most of today's quantum algorithms may not achieve practical speedups. Material science and chemistry have a huge potential and we hope more practical algorithms will be invented based on our guidelines
- Due to limitations of input and output bandwidth, quantum computers will be practical for "big compute" problems on small data, not big data problems.
- Quadratic speedups delivered by algorithms such as Grover's search are insufficient for practical quantum advantage without significant improvements across the entire software/hardware stack.



with an optimistic assumption for a hypothetical quantum computer that may be available in the next decades with 10,000 error-corrected logical qubits, 10 μ s gate time for logical operations, the ability to simultaneously perform gate operations on all qubits and all-to-all connectivity for fault tolerant two-qubit gates.^a

I/O bandwidth. We first consider the fundamental I/O bottleneck that limits quantum computers in their interaction with the classical world,

a Note that no quantum error correction scheme exists today that allows simultaneous execution of gates and all-to-all connectivity without at least a $O(\sqrt{N})$ slowdown for N qubits.

which determines bounds for data input and output bandwidths. Scalable implementations of quantum random access memory (QRAM^{8,9}) demand a fault-tolerant error corrected implementation and the bandwidth is then fundamentally limited by the number of quantum gate operations or measurements that can be performed per unit time. We assume only a single gate operation per input bit. For our optimistic future quantum computer, the resulting rate is 10,000-times smaller than for an existing classical chip (see Table 1). We immediately see that any problem limited by accessing classical data, such as search problems in databases, will be solved faster by classical

computers. Similarly, a potentially exponential quantum speedup in linear algebra problems¹² vanishes when the matrix must be loaded from classical data, or when the full solution vector should be read out. Generally, quantum computers will be practical for “big compute” problems on small data, not big data problems.

Crossover scale. With quantum speedup, asymptotically fewer operations will be needed on a quantum computer than on a classical computer. Due to the high operational complexity and slower gate operations, however, each operation on a quantum computer will be slower than a corresponding classical one. As sketched in the accompanying figure, classical computers will always be faster for small problems and quantum advantage is realized beyond a problem-dependent crossover scale where the gain due to quantum speedup overcomes the constant slowdown of the quantum computer. To have real practical impact, the crossover time must be short, not more than weeks. Constants matter in determining the utility for applications, as with any runtime estimate in computing.

Compute performance. To model performance, we employ the well-known work-depth model from classical parallel computing to determine upper bounds of classical silicon-based computations and an extension for quantum computations. In this model, the work is the total number of operations and applies to both classical and quantum executions. In Table 1, we provide concrete examples using three types of operations: logical operations, 16-bit floating point, and 32-bit integer or fixed-point arithmetic operations for numerical modeling. For the quantum costs, we consider only the most expensive parts in our estimates, again benefiting quantum computers; for arithmetic, we count just the dominant cost of multiplications, assuming additions are free. Furthermore, for floating point multiplication, we consider only the cost of the multiplication of the mantissa (10 bits in fp16). We ignore all further overheads incurred by the quantum algorithm due to reversible computations, as well as the significant cost of mapping to a specific

Quantum speedup.

The time needed to solve certain problems with quantum algorithms increases more slowly than that of any known classical algorithm as the problem size N increases. To be practical, however, we need more than an asymptotic speedup: the crossover time where quantum advantage gets realized needs to be reasonably short and the crossover problem size not too large. (For illustrative purposes, the time axis is scaled such that the quantum algorithm is a straight line.)

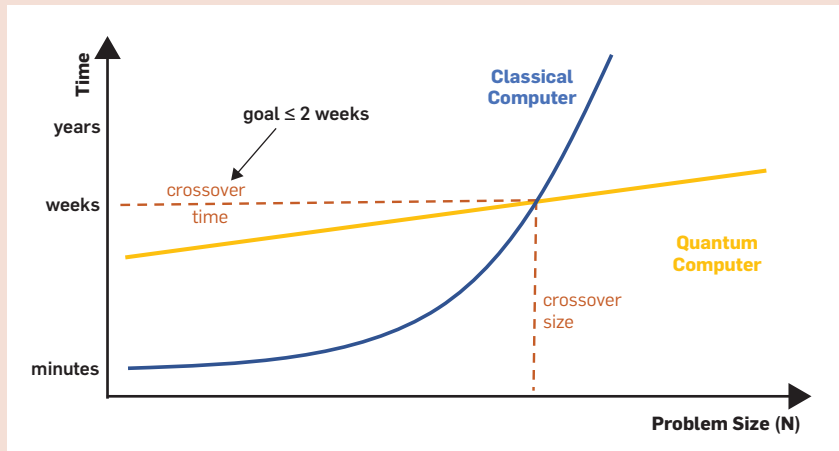


Table 1. Performance comparison.

We compare the peak performance of a single classical chip that can be manufactured today (like an NVIDIA A100 GPU, or an ASIC with a similar number of transistors) with a future quantum computer with 10,000 error-corrected logical qubits, 10 μ s gate time for logical operations and all-to-all connectivity. We consider an estimate of the I/O bandwidth (namely the number of operations per second) and three types of operations: logical binary operations, 16-bit floating point, 32-bit integer or fixed-point arithmetic multiply add operations.

	GPU	ASIC	Future Quantum
I/O Bandwidth	10,000 Gbit/s	10,000 G/s	1 Gbit/s
Operation throughput			
16-bit floating point	195 Top/s	550 Top/s	10.5 kop/s
32-bit integer	9.75 Top/s	215 Top/s	0.83 kop/s
binary (Boolean logical)	4,992 Top/s	77,000 Top/s	235 kop/s

hardware architecture with limited qubit connectivity.

Crossover times for classical and quantum computation. To estimate lower bounds for the crossover times, we consider that while both classical and quantum computers must evaluate the same functions (usually called oracles) that describe a problem, quantum computers require fewer evaluations thereof due to quantum speedup. At the root of many quantum acceleration proposals lies a quadratic quantum speedup, including the well-known *Grover algorithm*.^{10,11} For such an algorithm, a problem that needs X function calls on a quantum computer requires quadratically more, namely on the order of X^2 calls on a classical computer. To overcome the large constant performance difference between a quantum computer and a classical computer, which Table 1 shows to be more than a factor of 10^{10} , many function calls $X \gg 10^{10}$ is needed for the quantum speedup to deliver a practical advantage. In Table 2, we estimate upper bounds for the complexity of the function that will lead to a cross-over time of 10^6 seconds, or approximately two weeks.

We see that with quadratic speedup even a single floating point or integer operation leads to crossover times of several months. Furthermore, at most 68 binary logical operations can be afforded to stay within our desired crossover time of two weeks, which is too low for any non-trivial application. Keeping in mind that these estimates are pessimistic for classical computation (a single of today's classical chips) and overly optimistic for quantum computing (only considering the multiplication of the mantissa and assuming all-to-all qubit connectivity), we come to the clear conclusion that quadratic speedups are insufficient for practical quantum advantage. The numbers look better for cubic or quartic speedups where thousands or millions of operations may be feasible, and we conclude, similarly to Babbush et al.,³ that at least cubic or quartic speedups are required for a practical quantum advantage.

As a result of our overly optimistic assumptions in favor of quantum computing, these conclusions will remain valid even with significant advances

Table 2. Crossover operation counts for quantum algorithms with quadratic, cubic, and quartic speedups.

We determine the number of operations that can be afforded per function call (see the accompanying figure) for a quantum computer to show an advantage over a classical computer using a quantum algorithm with quadratic, cubic, and quartic quantum speedup. The number of oracle calls required to reach the crossover point with a quadratic, cubic, and quartic speedup is computed using the relative runtimes of a single oracle evaluation, and the total runtime of 10^6 seconds is then used to compute how many basic operations can be afforded in each oracle call. Since we make optimistic assumptions for a future quantum computer, we ignore overheads of reversible arithmetic for quantum computing and limit the classical computer to a single chip that can be manufactured today. The actual crossover operation counts will be significantly smaller. A similar analysis for quantum algorithms with exponential speedups yields promising operation budgets for all datatypes.

Operation type	Maximum number of operations for practical		
	quadratic speedup	cubic speedup	quartic speedup
16-bit floating point	0.2	45,800	2,800,000
32-bit integer	0.003	1,630	130,000
Binary (logical)	68	12,500,000	712,000,000

in quantum technology of multiple orders of magnitude.

Practical and impractical applications. We can now use these considerations to discuss several classes of applications where our fundamental bounds draw a line for quantum practicality. The most likely problems to allow for a practical quantum advantage are those with exponential quantum speedup. This includes the simulation of quantum systems for problems in chemistry, materials science, and quantum physics, as well as cryptanalysis using Shor's algorithm.¹⁶ The solution of linear systems of equations for highly structured problems¹² also has an exponential speedup, but the I/O limitations discussed above will limit the practicality and undo this advantage if the matrix has to be loaded from memory instead of being computed based on limited data or knowledge of the full solution is required (as opposed to just some limited information obtained by sampling the solution).

Equally important, we identify likely dead ends in the maze of applications. A large range of problem areas with quadratic quantum speedups, such as many current machine learning training approaches, accelerating drug design and protein folding with Grover's algorithm, speeding up Monte Carlo simulations through quantum walks, as well as more traditional scientific computing simulations including the solution of many non-linear

systems of equations, such as fluid dynamics in the turbulent regime, weather, and climate simulations will not achieve quantum advantage with current quantum algorithms in the foreseeable future. We also conclude that the identified I/O limits constrain the performance of quantum computing for big data problems, unstructured linear systems, and database search based on Grover's algorithm such that a speedup is unlikely in those cases. Furthermore, Aaronson et al.¹ show the achievable quantum speedup of unstructured black-box algorithms is limited to $O(N^4)$. This implies that any algorithm achieving higher speedup must exploit structure in the problem it solves.

These considerations help with separating hype from practicality in the search for quantum applications and can guide algorithmic developments. Specifically, our analysis shows it is necessary for the community to focus on super-quadratic speedups, ideally exponential speedups, and one needs to carefully consider I/O bottlenecks when deriving algorithms to exploit quantum computation best. Therefore, *the most promising candidates for quantum practicality are small-data problems with exponential speedup*. Specific examples where this is the case are quantum problems in chemistry and materials science,⁵ which we identify as the most promising application. We recommend using precise requirements models⁴ to get

more reliable and realistic (less optimistic) estimates in cases where our rough guidelines indicate a potential practical quantum advantage.

Methods

Here, we provide more details for how we obtained the numbers mentioned earlier. We compare our quantum computer with a single microprocessor chip like the one used in the NVIDIA A100 GPU.¹⁵ The A100 chip is around 850mm^2 in size and manufactured in TSMC's 7nm N7 silicon process. A100 shows that such a chip fits around 54.2 billion transistors and can operate at a cycle time of around 0.7ns.

Determining peak operation throughputs. In Table 1, we provide concrete examples using three types of operations: logical operations, 16-bit floating point, and 32-bit integer arithmetic operations for numerical modeling. Other datatypes could be modeled using our methodology as well.

Classical NVIDIA A100. According to its datasheet, NVIDIA's A100 GPU, a SIMT-style von Neumann load store architecture, delivers 312 tera-operations per second (Top/s) with half precision floating point (fp16) through tensor cores and 78Top/s through the normal processing pipeline. NVIDIA assumes a 50/50 mix of addition and multiplication operations and thus, we divide the number by two, yielding 195Top/s fp16 performance. The datasheet states 19.5Top/s for 32-bit integer operations, again assuming a 50/50 mix of addition and multiplication, leading to an effective 9.75Top/s. The binary tensor core performance is listed as 4,992Top/s with a limited set of instructions.

Classical special-purpose ASIC. Our main analysis assumes that we build a special-purpose ASIC using a similar technology. If we were to fill the equivalent chip-space of an A100 with a specialized circuit, we would use existing execution units, for which the size is typically measured in gate equivalents (GE). A 16-bit floating point unit (FPU) with addition and multiplication functions requires approximately 7kGE, a 32-bit integer unit requires 18kGE,¹⁴ and we assume 50GE for a simple binary operation. All units include operand buffer registers and support a set of programmable instructions. We note that simple addition or multipli-

Our analysis shows a wide range of often-cited applications is unlikely to result in a practical quantum advantage without significant algorithmic improvements.

cation circuits would be significantly cheaper. If we assume a transistor-to-gate ratio of 10^{13} and that 50% of the total chip area is used for control logic of a dataflow ASIC with the required buffering, we can fit $54.2B/(7k \cdot 10 \cdot 2) = 387k$ fp16 units. Similarly, we can fit $54.2B/(18k \cdot 10 \cdot 2) = 151k$ int32, or $54.2B/(50 \cdot 10 \cdot 2) = 54.2M$ bin2 units on our hypothetical chip. Assuming a cycle time of 0.7ns, this leads to a total operation rate of 0.55 fp16, 0.22 int32, and 77.4 bin Pop/s for an application-specific ASIC with the A100's technology and budget. The ASIC thus leads to a raw speedup between approximately 2x and 15x over a programmable circuit. Thus, on classical silicon, the performance ranges approximately between 10^{13} and 10^{16} op/s for binary, int32, and fp16 types.

Hypothetical future quantum computer. To determine the costs of N -bit multiplication on a quantum computer, we choose the controlled adder from Gidney⁶ and implement the multiplication using N single-bit controlled adders, each requiring $2N$ CCZ magic states. These states are produced in so called "magic state factories" that are implemented on the physical chip. While the resulting multiplier is entirely sequential, we found that this construction allows for more units to be placed on one chip than for a low-depth adder and/or for a tree-like reduction of partial products since the number of CCZ states is lower (and thus fewer magic state factories are required), and the number of work-qubits is lower. The resulting multiplier has a CCZ-depth and count of $2N^2$ using $5N - 1$ qubits ($2N$ input, $2N - 1$ output, N ancilla for the addition).

To compute the space overhead due to CCZ factories, we first use the analysis of Gidney and Fowler⁷ to compute the number of physical qubits per factory when aiming for circuits (programs) using $\approx 10^8$ CCZ magic states with physical gate errors of 10^{-3} . We approximate the overhead in terms of logical qubits by dividing the physical space overhead by $2d^2$, where we choose the error-correcting code distance $d = 2 \cdot 31^2$ to be the same as the distance used for the second level of distillation.⁷ Thus we divide Gidney and Fowler's 147,904 physical qubits per factory (for details consult the an-

cillary spreadsheet (field B40) of Gidney and Fowler) by $2d^2 = 2 \cdot 31^2$ and get an equivalent space of 77 logical qubits per factory.

For the multiplier of the 10-bit mantissa of an fp16 floating point number, we need $2 \cdot 10^2 = 200$ CCZ states and $5 \cdot 10 = 50$ qubits. Since each factory takes 5.5 cycles⁷ and we can pipeline the production of CCZ states, we assume 5.5 factories per multiplication unit such that multipliers do not wait for magic state production on average. Thus, each multiplier requires 200 cycles and $5N + 5.5 \cdot 77 = 50 + 5.5 \cdot 77 = 473.5$ qubits. With a total of 10,000 logical qubits, we can implement 21 10-bit multipliers on our hypothetical quantum chip. With 10 μ s cycle time, the 200-cycle latency, we get the final rate of less than 10^5 cycle/s / (200 cycle/op) $\cdot 21 = 10.5$ kop/s. For int32 ($N=32$), the calculation is equivalent. For binary, we assume two input and one output qubit for the (binary) adder (Toffoli gate) which does not need ancillas. The results are summarized in Table 1.

A note on parallelism. We assumed massively parallel execution of the oracle on both the classical and quantum computer (that is, oracles with a depth of one). If the oracle does not admit such parallelization, for example, if depth = work in the worst-case scenario, then the comparison becomes more favorable towards the quantum computer. One could model this scenario by allowing the classical computer to only perform one operation per cycle. With a 2GHz clock frequency, this would mean a slowdown of about 100,000 times for fp16 on the GPU. In this *extremely unrealistic* algorithmic worst case, the oracle would still have to consist of only several thousands of fp16 operations with a quadratic speedup. However, we note that in practice, most oracles have low depth and parallelization across a single chip is achievable, which is what we assumed.

Determining maximum operation counts per oracle call. In Table 2, we list the maximum number of operations of a certain type that can be run to achieve a quantum speedup within a runtime of 10^6 seconds (a little more than two weeks). The maximum number of classical operations that can be performed with a single classical chip in 10^6 seconds would be: 0.55 fp16, 0.22 int32,

and 77.4 bin Zop. Similarly, assuming the rates from Table 1, for a quantum chip: 7, 4, 2, and 350 Gop, respectively.

We now assume that all calculations are used in oracle calls on the quantum computer and we ignore all further costs on the quantum machine. We start by modeling algorithms that provide polynomial X^k speedup, for small constants k . For example, for Grover's algorithms,¹¹ $k + 2$. It is clear quantum computers are asymptotically faster (in the number of oracle queries) for any $k > 1$. However, we are interested to find the oracle complexity (that is, the number of operations required to evaluate it) for which a quantum computer is faster than a classical computer within the time-window of 10^6 seconds.


Let the number of operations required to evaluate a single oracle call be M and let the number of required invocations be N . It takes a classical computer time $T_c = N^k \cdot M \cdot t_c$, whereas a quantum computer solves the same problem in time $T_q = N^k \cdot M \cdot t_q$ where t_c and t_q denote the time to evaluate an operation on a classical and on a quantum computer, respectively. By demanding that the quantum computer should solve the problem faster than the classical computer and within 10^6 seconds, we find

$$k^{-1} \sqrt[k]{\frac{t_q}{t_c}} \leq N \leq \frac{10^6}{t_q \cdot M^k}$$

which allows us to compute the maximal number of basic operations per oracle evaluation such that the quantum computer still achieves a practical speedup:

$$M \leq 10^6 \cdot k^{-1} \sqrt[k]{\frac{t_c}{t_q}}$$

Determining I/O bandwidth. We use the I/O bandwidth specified in NVIDIA's A100 datasheet for our classical chips or the quantum computer, we assume that one quantum gate is required per bit of I/O. Using all 10,000 qubits for reading/writing, this yields an estimate of the I/O bandwidth $B \approx \frac{10,000}{10^{-3}} = 1$ Gbit/s.

Acknowledgments. We thank L. Benini for helpful discussions about ASIC and processor design and related overheads and W. van Dam and anonymous reviewers for comments that improved an earlier draft. 

References

1. Aaronson, S., Ben-David, S., Kothari, R., Rao, S. and Tal, A. Degree vs. approximate degree and quantum implications of Huang's sensitivity theorem. In *Proceedings of the 53rd Annual ACM SIGACT Symp. Theory of Computing*, 2021, 1330–1342, ACM, New York, NY, USA.
2. Arute, F. et al. Quantum supremacy using a programmable superconducting processor. *Nature* 574 (2019), 505–510.
3. Babbush, R., McClean, J.R., Newman, M., Gidney, C., Boixo, S., and Neven, H. Focus beyond quadratic speedups for error-corrected quantum advantage. *PRX Quantum*, 2:010103, Mar 2021.
4. Beverland, M.E. et al. Assessing requirements to scale to practical quantum advantage, 2022.
5. Feynman, R.P. Simulating physics with computers. *Intern. J. Theoretical Physics* 21, 6/7 (1981).
6. Gidney, C. Halving the cost of quantum addition. *Quantum* 2, 74 (June 2018).
7. Gidney, C. and Fowler, A.G. Efficient magic state factories with a catalyzed CCZ to 2 T transformation. *Quantum* 3, 135 (Apr. 2019).
8. Giovannetti, V., Lloyd, S., and Maccone, L. Quantum random access memory. *Physical Review Letters* 100, 16 (Apr. 2008).
9. Giovannetti, V., Lloyd, S., and Maccone, L. Architectures for a quantum random access memory. *Physical Review A* 78, 5 (Nov. 2008).
10. Grover, L. K. Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters* 79, 2 (1997).
11. Grover, L. K. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symp. Theory of Computing*, 1996. ACM, New York, NY, 212–219.
12. Harrow, A.W., Hassidim, A. and Lloyd, S. Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* 103, 150502 (Oct. 2009).
13. Jones, S. GlobalFoundries: 7nm, 5nm and 3nm Logic, current and projected processes, 2018; <https://bit.ly/3XE3ucJ>.
14. Mach, S., Schuiki, F., Zaruba, F. and Benini, L. Fpnew: An open-source multiformat floating-point unit architecture for energy-proportional transprecision computing. *IEEE Trans. Very Large-Scale Integration* 29, 4 (2021), 774–787.
15. NVIDIA Corp. NVIDIA A100 Tensor Core GPU Architecture, 2020; <https://bit.ly/3WjVc91>.
16. Shor, P.W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* 26, 5 (Oct. 1997), 1484–1509.

Torsten Hoefler is Consulting Researcher at Microsoft Corporation, Redmond, WA, USA, and a professor at ETH Zurich, Switzerland.

Thomas Häner is Research Scientist at Amazon Web Services, Zurich, Switzerland. This work was done when he worked at Microsoft, Zurich, prior to his joining AWS.

Matthias Troyer is Technical Fellow and Corporate Vice President at Microsoft, Redmond, WA, USA.

Copyright held by authors.
Publication rights licensed to ACM.



Watch the authors discuss this work in the exclusive *Communications* video. <https://caom.acm.org/videos/quantum-advantage>