Focus beyond quadratic speedups for error-corrected quantum advantage

Ryan Babbush,[1,*] Jarrod R. McClean,[1,†] Craig Gidney,[1] Sergio Boixo,[1] and Hartmut Neven[1]

[1] *Google Quantum AI Team, Venice, CA 90291, United States of America*

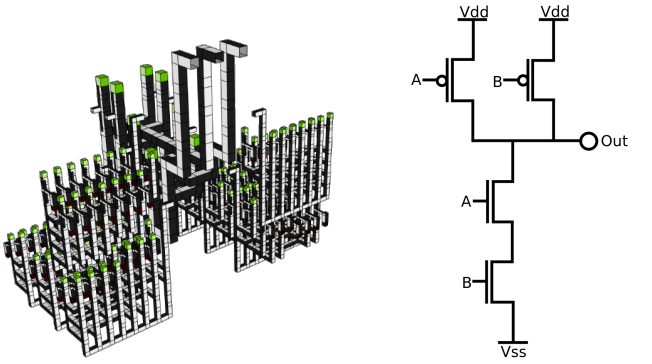
(Dated: November 10, 2020)

We discuss conditions under which it would be possible for a modest fault-tolerant quantum computer to realize a runtime advantage by executing a quantum algorithm with only a small polynomial speedup over the best classical alternative. The challenge is that the computation must finish within a reasonable amount of time while being difficult enough that the small quantum scaling advantage would compensate for the large constant factor overheads associated with error-correction. We compute several examples of such runtimes using state-of-the-art surface code constructions for superconducting qubits under a variety of assumptions. We conclude that quadratic speedups will not enable quantum advantage on early generations of such fault-tolerant devices unless there is a significant improvement in how we would realize quantum error-correction. While this conclusion persists even if we were to increase the rate of logical gates in the surface code by more than an order of magnitude, we also repeat this analysis for speedups by other polynomial degrees and find that quartic speedups look significantly more practical.

Introduction

One of the most important goals of the field of quantum computing is to eventually build a fault-tolerant quantum computer. But what valuable and classically challenging problems could we actually solve on such a device? Among the most compelling applications are quantum simulation [1, 2] and prime factoring [3]. Quantum algorithms for these tasks give exponential speedups over known classical alternatives but would have limited impact compared to significant improvements in our ability to address problems in broad areas of industrial relevance such as optimization and machine learning. However, while quantum algorithms exist for these applications, the most rigorous results have only been able to show a large speedup in contrived settings or a smaller speedup across a broad range of problems. For example, many quantum algorithms (often based on amplitude amplification [4]) give quadratic speedups for tasks such as search [5], optimization [5–7], Monte Carlo [4, 8, 9] various areas of machine learning [10, 11] and more. However, attempts [7, 12] to assess the overheads of some such applications within fault-tolerance have come up with discouraging predictions for what would be required to achieve practical advantage against classical algorithms.

The central issue is that quantum error-correction and device operation time introduce significant constant factor slowdowns to the algorithm runtime (see Figure 1). These large overheads present many challenges for the practical realization of useful fault-tolerant devices. However, for applications that benefit from an exponential speedup relative to classical algorithms, the exponential scaling of the classical approach quickly catches up to the large constant factors of the quantum approach so that one can achieve a practical runtime advantage for even modest problem sizes. This is borne out through numerous studies on the cost of error-correcting applications with exponential scaling advantage in areas such as quantum chemistry [14–16], quantum simulation of lattice models [17, 18] and prime factoring [19].



(a) "Quantum NAND" > 10 qubitseconds

(b) "Classical NAND" $< 10^{-9}$ transistorseconds

FIG. 1. The primary obstacle in realizing a runtime advantage for low degree quantum speedups is the enormous slowdown when performing basic logic operations within quantum error-correction. (a) A surface code Toffoli factory for distilling Toffoli gates (which act as the NAND gate when the target bit is ON) requires spacetime volume greater than ten qubit-seconds under reasonable assumptions on the capabilities of an error-corrected superconducting qubit platform [13]. (b) A NAND circuit realized in CMOS can be executed with just a few transistors in well under a nanosecond. Thus, there is roughly a ten order of magnitude difference between the spacetime volume required for comparable operations on an error-corrected quantum computer and a classical computer.

In this perspective we discuss when it would be practical for a modest fault-tolerant quantum computer to realize a quantum advantage with quantum algorithms giving only a small polynomial speedup over their classical competition. We will see that with only a low order (e.g., quadratic) speedup, exorbitantly long runtimes are

* Corresponding author: babbush@google.com
† Corresponding author: jmcclean@google.com

sometimes required in order for the slightly worse scaling of the classical algorithm to catch up to the slightly better scaling (but worse constant factors) of the quantum algorithm. We will argue that the problem is especially pronounced when the best classical algorithms for a problem can also be easily parallelized.

Our analysis will emphasize current projections within the surface code [20] since it has the highest threshold error rate for a two-dimensional quantum computing architecture and is generally regarded as the most practical quantum error-correcting code [21]. We will focus on a modest realization of the surface code that would involve enough resources to perform classically intractable calculations but only support a few state distillation factories. Our analysis differs from results such as [7, 12] by addressing the prospects for achieving quantum advantage via polynomial speedup for a broad class of algorithms, rather than for specific problems. We will assume that there is some problem which can be solved by a classical computer that makes M^d calls to a "classical primitive" circuit or by a quantum computer which makes M calls to a "quantum primitive" circuit (which is often, but not always, related to the classical primitive circuit). This corresponds to an order d polynomial quantum speedup in the number of queries to these subroutines. For $d = 2$, this is especially evocative of a common class of quantum algorithms leveraging amplitude amplification. This generously assumes no prefactor overhead in a quantum implementation of an algorithm with respect to the number of calls required and along with other crude assumptions, allows us to bound the crossover time.

Our back-of-the-envelope analysis makes many assumptions which are overly optimistic towards the quantum computer and yet we still conclude that the prospects look poor for quadratic speedups with current error-correcting codes and architectures to outperform classical computers in time-to-solution. It seems that to realize a quantum advantage with reasonable fault-tolerant resources, one must either focus beyond quadratic speedups, dramatically improve techniques for error-correction, or do both. An encouraging finding is that the prospects for error-corrected quantum advantage look significantly better with quartic speedups. Of course, there might exist use cases involving quadratic speedups that defy the framework of this analysis. Either way, we hope this perspective will encourage the field to critically examine the prospects for quantum advantage with error-corrected quadratic speedups and either produce examples where it is feasible, or focus more effort on algorithms with larger speedups.

Relationship between primitive times and runtime

Many quantum algorithms are built on coherent access to primitives implemented with classical logic. For example, this classical logic might be required to compute the value of a classical cost function for optimization [12], to evaluate a function of a trajectory of some security that one is pricing with Monte Carlo [9], or to compute some classical criteria that flags a marked state for which one might be searching [5]. We will define the runtime of the quantum and classical algorithms as

$$\mathcal{T}_Q = M\, t_Q \qquad \mathcal{T}_C = M^d\, t_C \tag{1}$$

where $\mathcal{T}$ gives the total runtime of the algorithm, M is the number of primitive calls required, d is the order of the polynomial speedup the quantum computer achieves and t is the time required to perform a call. Throughout this perspective the subscripts Q and C will denote "quantum" and "classical" implementations.

The condition for quantum advantage is

$$\mathcal{T}_Q < \mathcal{T}_C \qquad \text{and thus,} \qquad M > \left(\frac{t_Q}{t_C}\right)^{\frac{1}{d-1}}. \tag{2}$$

We see then that whenever a problem will require enough calls M that a quantum advantage is possible,

$$\mathcal{T}_Q > \mathcal{T}^\star \equiv t_Q \left(\frac{t_Q}{t_C}\right)^{\frac{1}{d-1}} \tag{3}$$

where $\mathcal{T}^\star$ is the "breakeven time" which occurs when $\mathcal{T}_Q = \mathcal{T}_C$, corresponding to onset of quantum advantage. As emphasized in Figure 1, we will see that the fundamental challenge in realizing this runtime advantage against classical computers (for small d) is that $t_Q \gg t_C$ in error-corrected contexts, making $\mathcal{T}^\star$ very large.

Rather than use a single CPU for the classical approach, one might instead parallelize the algorithm using P classical CPUs. This will reduce the total classical runtime to

$$\mathcal{T}_C = \frac{M^d\, t_C}{S} \qquad S = \left(\alpha + \frac{1-\alpha}{P}\right)^{-1} \tag{4}$$

where α is the fraction of the algorithm which must be executed in serial and S is the speedup factor due to parallelization consistent with the "Amdahl's law" [22]. Note that Amdahl's law scaling is considered somewhat pessimistic as one can often adjust the size of problems to fully exploit the computing power that becomes available with more parallelism (e.g., see "Gustafson's law" [23] for a more optimistic formula for S). But it also seems that in most situations where one might hope to find a quadratic speedup with a quantum computer (e.g. applications such as search, optimization, Monte Carlo, regression, etc.) the corresponding classical approach is embarrassingly parallel (suggesting that α is small enough that $S \approx P$ for reasonable values of P). Regardless of the form of S, classical parallelism leads to the following revised conditions for quantum advantage:

$$M > \left(\frac{t_Q S}{t_C}\right)^{\frac{1}{d-1}} \qquad \text{and} \qquad \mathcal{T}^\star = t_Q \left(\frac{t_Q S}{t_C}\right)^{\frac{1}{d-1}}. \tag{5}$$

While parallel efficiency might be limited for some applications, any implementation of an error-correcting code will also require substantial classical co-processing in order to perform decoding, and this is likely to require thousands of classical cores. Although many quantum algorithms can also benefit from various forms of parallelism, we are considering an early fault tolerance setting where there is likely an insufficient number of logical qubits to exploit a space-time tradeoff to the same extent.

Implementing error-corrected quantum primitives

We will now explain the principle overheads believed to be required for the best candidate for quantum error-correction on a two-dimensional lattice: the surface code. Toffoli gates are required to implement classical logic on a quantum computer but cannot be implemented transversely within practical implementations of the surface code. Instead, one must implement these gates by first distilling resource states. In particular, to implement a Toffoli gate one requires a CCZ state ($|\mathrm{CCZ}\rangle = \mathrm{CCZ}\,|+++\rangle$) and these states are consumed during the implementation of the gate. Distilling CCZ states requires a substantial amount of both time and hardware and thus, they are usually the bottleneck in realizing quantum algorithms within the surface code.

Here, we will focus on the state-of-the-art Toffoli factory constructions of [13] which are based on applying the lattice surgery constructions of [24] to the fault-tolerant Toffoli protocols of [25, 26]. Using that approach one Toffoli gate requires $5.5 \times d$ surface code cycles, where d is the code distance. The time per round of the surface code, including decoding time is expected to be around $1\,\mu s$ in superconducting qubits. Our analysis will assume a code distance in the vicinity of $d = 30$. This would be sufficient for an algorithm with billions of gates and physical gate error rates on the order of 10^{-3} (as our analysis will reveal, even more than a billion gates would likely be required to obtain quantum advantage with a modest polynomial speedup). With these assumptions, our model predicts a Toffoli gate time of

$$t_G = 30 \times 5.5 \times 1\,\mu\mathrm{s} \approx 170\,\mu\mathrm{s}\,. \tag{6}$$

This rough approximation matches the more detailed resource estimate of Ref. [13]. We discuss these estimates in more detail in Appendix A. Note that a potential ion trap physical gates are at least $100\times$ slower than superconducting qubits. Therefore, even assuming that they operate physical gates in parallel with high fidelity, an ion trap surface code implementation of a Toffoli gate would take time on the order of $t_G \simeq 17\,\mathrm{ms}$.

Under the aforementioned assumptions which are specific to contemporary realizations of the surface code using superconducting qubits we could express the quantum primitive runtime as

$$t_Q = t_G \cdot G = 170\,\mu\mathrm{s} \cdot G\,, \tag{7}$$

where G is the number of Toffoli gates required to implement the quantum primitive. On a very large surface code quantum computer one could instead use multiple Toffoli factories (at a high cost in the number of physical qubits required) in order to reduce t_Q by performing state distillation in parallel. However, the Toffoli gates are only about two orders of magnitude slower than the Clifford gates and when using multiple factories one needs to account for routing overhead. Thus, while t_Q can be reduced at the cost of using many more qubits, only by a factor that is between about ten and one-hundred.

If N is the number of qubits on which this problem is defined then a sensible lower bound would seem to be $G \geq N$ and thus, $t_Q \geq 170\,\mu\mathrm{s} \cdot N$. For example, in Grover's algorithm [5] one must perform a reflection that requires $\mathcal{O}(N)$ Toffoli gates. In order to achieve a quantum advantage we would need to focus on problem sizes that are sufficiently large that enough calls can be made so that Eq. (2) is satisfied. We find it difficult to imagine satisfying this condition for problem sizes less than one-hundred qubits. Thus, an approximate "lower bound" (using $N = 100$) would be

$$t_Q \geq 17\,\mathrm{ms}\,. \tag{8}$$

In addition to this lower bound, we will also consider a specific, realistic example to keep our estimates grounded. We will focus on the quantum accelerated simulated annealing by qubitized quantum walk algorithm studied in [27, 28], which appears to provide a quadratic speedup over classical simulated annealing (at least in terms of the best known bounds) in terms of the mixing time of the Markov chain under certain assumptions [29]. This is among the most efficient algorithms compiled in [12] and for the Sherrington-Kirkpatrick model [30], the implementation complexity is $5N + \mathcal{O}(\log N)$ (neglecting some subdominant scalings that depend on precision), which is only worse than the scaling of our lower bound by a factor of five. For example, for an $N = 512$ qubit instance, the work of [12] shows that only about $2.6{\times}10^3$ Toffoli gates are required to make an update. Thus, for that problem size (which we choose to facilitate a comparison to classical algorithms that we will discuss later) we have that

$$t_Q = 440\,\mathrm{ms}\,. \tag{9}$$

Implementing classical primitives

Classical computers are very fast; a typical 3 GHz CPU can perform several billion 64 bit operations (e.g., floating point multiplications) per second. We might crudely write that the classical primitive time is $t_C = 330\,\mathrm{ps} \cdot L$ where L is the number of classical clock cycles required to implement the classical primitive. For our first example comparison of quantum and classical primitives we will assume that any classical logic operation that would

require one Toffoli in the quantum primitive can be executed during one classical clock cycle in the classical primitive. This seems generous to the quantum computer since many operations that would take a single clock cycle on a classical computer would actually require thousands of Toffolis. (Note that we are not assuming any scaling advantage for the quantum computer in the primitive implementations.) One might worry about memory-bound classical primitives (since calls to main memory can take hundreds of clock cycles) but since problems defined on more than thousands of logical bits would be infeasible to process on a small fault-tolerant quantum computer we expect that the memory required for the corresponding classical primitives can be held in cache.

Thus, a corresponding bound on the time to realize a classical primitive for a problem where a quantum computer could realize a quantum primitive with anywhere near the lower bound time given in the prior section ($t_Q \geq 170\,\mu\text{s} \cdot N$) is $t_C \leq 330\,\text{ps} \cdot N$, and for $N = 100$,

$$t_C \leq 33\,\text{ns}\,. \tag{10}$$

Even though the equivalence we make between Toffolis and classical compute cycles is seemingly generous to the quantum computer, the assumption of such a cheap primitive on the quantum side (only 100 Toffolis) results in what appears to be a fairly cheap primitive on the classical side. However, because Eq. (5) scales worse with t_Q than with t_C, this assumption is ultimately optimistic towards the overall crossover time.

Consistent with the prior section, we will also discuss the classical primitive time required to apply simulated annealing to an instance of the Sherrington-Kirkpatrick model. Using the techniques developed in [31], a performant implementation of classical simulated annealing code for an $N = 512$ instance of the Sherrington-Kirkpatrick model can perform a simulated annealing step in roughly 7 CPU-nanoseconds [12] (this accounts for the fact that most updates for the Sherrington-Kirkpatrick model are rejected); thus in that case,

$$t_C = 7\,\text{ns}. \tag{11}$$

But given the high costs of quantum computing it is unclear that we should compare to a single classical core.

Minimum runtime for quadratic quantum advantage

Here we discuss the ramifications that the primitive runtimes discussed in the prior two sections have for the minimum time to achieve advantage according to Eq. (3) in the case of a quadratic quantum speedup. Quadratic speedups are ubiquitous in quantum computing and appear in many contexts such as Grover's algorithm [5] (e.g., used for search) and its generalization, amplitude amplification [4] (e.g., used in Monte Carlo), as well as many other contexts such as optimization.

First, we will compare the example of a quantum primitive requiring only $N = 100$ Toffolis and $t_Q = 17\,\text{ms}$. We argued that any such primitive could likely be computed in $t_C = 33\,\text{ns}$ on a single core. For this example, $\mathcal{T}^\star = t_Q^2/t_C = 2.4$ hours. One might object to this minimal example on the grounds that it seems unlikely any interesting primitive would require only 100 Toffolis. While this is true, we point out that because quantum runtime is quadratic in the quantum primitive time and only inversely proportional the classical primitive time, the overall crossover time can only get worse by assuming that more than 100 Toffolis would be required.

Next, we will compare to the example of quantum accelerated simulated annealing. We focus on this example because the steps of the quantum algorithm have been concretely compiled, appear quite efficient, and have a clear classical analogue. Here, for an $N = 512$ qubit instance we have that $t_Q^2/t_C = 320\,\text{days}$, reproducing the finding in [12]. We note that quantum advantage in this case would occur when $M > t_Q/t_C = 6.3 \times 10^7$. This means that 4.0×10^{15} calls would need to be required for the classical algorithm. However, most $N = 512$ Sherrington-Kirkpatrick model instances would require many fewer calls to solve with classical simulated annealing and so one would need to focus on an even bigger system for which the numbers will look yet worse for the quantum computer. Notice that our simulated annealing example gave a quantum runtime that is much longer than the resources required for the quantum primitive with $N = 100$ Toffolis. This is because the notion that it would take a classical computer an entire clock cycle to do what a quantum computer could accomplish with a single Toffoli is very generous to the quantum computer.

At first glance, the quantum runtime of 2.4 hours to achieve advantage for the primitive with just 100 Toffolis seems encouraging. Unfortunately, this was just for a single classical core. Even most laptops have on the order of ten cores these days and again, most of the problems where quantum computers display a quadratic advantage are classically embarrassingly parallel problems. Furthermore, error-corrected quantum computers are likely to use thousands of classical CPUs just for decoding. When using P different classical CPUs in parallel then the breakeven time is given by Eq. (5). Using that equation, if we take $P = 3{,}000$ CPUs for the classical task (rather than using them for error-correction), and if the classical algorithm is sufficiently parallelizable ($\alpha^{-1} \ll P$ so $S \approx P$), we see that the breakeven time even in this still quantum-generous example becomes one year. As we discuss in the next section there are also ways of parallelizing the quantum computations; e.g., by using multiple quantum computers or distillation factories.

The viability of higher polynomial speedups and the impact of faster error-correction

We report values of both M and $\mathcal{T}^\star$ assuming quantum speedups by different polynomial degrees under different amounts of classical parallelism in Table I. While the vi-

polynomial degree d	parallelism speedup S	resource "lower bound" iterations M	resource "lower bound" runtime $\mathcal{T}^\star$	simulated annealing iterations M	simulated annealing runtime $\mathcal{T}^\star$
Quadratic, $d = 2$	1	5.2×10^5	2.4 hours	6.3×10^7	320 days
	10^3	5.2×10^8	100 days	6.3×10^{10}	880 years
	10^6	5.2×10^{11}	280 years	6.3×10^{13}	880 millennia
Cubic, $d = 3$	1	7.2×10^2	12 seconds	7.9×10^3	58 minutes
	10^3	2.3×10^4	6.4 minutes	2.5×10^5	1.3 days
	10^6	7.2×10^5	3.4 hours	7.9×10^6	40 days
Quartic, $d = 4$	1	8.0×10^1	1.4 seconds	4.0×10^2	2.9 minutes
	10^3	8.0×10^2	14 seconds	4.0×10^3	29 minutes
	10^6	8.0×10^3	2.3 minutes	4.0×10^5	4.9 hours

TABLE I. Resources required to achieve quantum advantage assuming speedups of various polynomial degrees, d. We make this comparison against an adversary using distributed classical computing resources that achieve a speedup factor S and report the number of algorithm steps M and total runtime $\mathcal{T}^\star$ before a quantum speedup is possible. We make this comparison for both the informal resource "lower bound" we argued for in the text (using $t_Q \geq 17\,\text{ms}$ and $t_C \leq 33\,\text{ns}$), and for the specific example of quantum simulated annealing applied to the Sherrington-Kirkpatrick model using the quantum and classical implementations discussed in [12, 31] (giving $t_Q = 440\,\text{ms}$ and $t_C = 7\,\text{ns}$).

ability of quantum advantage with cubic speedups is still a bit ambiguous, the prospects of achieving quantum advantage given a quartic speedup are promising. Even the simulated annealing example run with a classical adversary with $S = 10^6$ parallelism would give quantum advantage after five hours of runtime if we assume a quartic speedup (while we do not expect a quartic speedup in that case, the comparison is still instructive).

It is rather surprising just how much of a difference there is for this example between assuming a quadratic speedup (requiring 880 millennia of runtime for advantage) and a quartic speedup (requiring just 4.9 hours of runtime for advantage). There are not as many examples of quartic speedups in quantum computing; however, perhaps if one can find a practical situation in which the quartic query complexity reductions of Ambainis *et al.* [32] and Aaronson *et al.* [33] can be realized, those could lead to a practical advantage in the surface code. We also expect that certain applications of quantum algorithms for linear systems [34] (such as for solving linear differential equations in high dimension) might lead to modest polynomial speedups higher than quadratic. It is also possible that some heuristic quantum algorithms for optimization might give larger than quadratic improvements, although this is still speculative.

Another question we might ask is, what happens if we were somehow able to implement Toffoli gates much faster in the surface code? For example, we might achieve this by fanning out and using more physical qubits per factory, more Toffoli factories, by inventing significantly more efficient protocols for Toffoli state distillation, or even by switching to a different technology with an intrinsically faster cycle time. We will perform this analysis for the case of quadratic speedups; there, the quantum runtime is reduced to $\mathcal{T}_Q = M\,t_Q/R$ where $R \geq 1$ is a speedup factor corresponding to performing Toffoli distillation in time $170\,\mu\text{s}/R$. In analogy to Eq. (5) this leads to the equations for a quadratic quantum speedup

$$M > \frac{t_Q S}{t_C R} \qquad \text{and} \qquad \mathcal{T}^\star = \frac{t_Q^2 S}{t_C R^2}\,. \tag{12}$$

In Table II we compute Eq. (12) for our example problems with $R = 10$, $R = 10^2$ and $R = 10^3$, assuming a classical adversary capable of achieving an $S = 10^3$ parallelism. We restrict ourselves to $S = 10^3$ due to the general difficulty in achieving high parallel efficiency described by Amdahl's law. However, note that for simulated annealing we can achieve $S = 10^6$ in practice (and so these numbers are overly optimistic for that case).

Unfortunately, even if Toffoli distillation rates improve by an order of magnitude it would not be enough to make quantum advantage with a quadratic speedup viable. If Toffoli distillation rates improve by two orders magnitude (making them essentially as cheap as Clifford gates) then it would still be challenging to obtain quantum advantage with a quadratic speedup (it would take more than a month for the simulated annealing example despite limiting the classical parallelism to $S = 10^3$) but we cannot categorically rule it out for all algorithms. At

speedup factor	resource "lower bound" iterations M	resource "lower bound" runtime $\mathcal{T}^\star$	simulated annealing iterations M	simulated annealing runtime $\mathcal{T}^\star$
$R = 10^1$	5.2×10^7	1.0 day	6.3×10^9	8.8 years
$R = 10^2$	5.2×10^6	15 minutes	6.3×10^8	32 days
$R = 10^3$	5.2×10^5	8.8 seconds	6.3×10^7	7.7 hours

TABLE II. Resources required to achieve quantum advantage under a quadratic speedup assuming Toffoli distillation time of $170\,\mu\text{s}/R$ and a classical adversary making use of classical parallelism with $S = 10^3$. The speedup factor R can account for improvements in error-correction implementations or in our estimates of their overheads. For example, $R = 10$ could be reached by using ten Toffoli factories if routing were very efficient (at the cost of requiring many more qubits).

three orders of magnitude speedup the story would be materially different but this would likely require a significant breakthrough. Even if classical processing and signal propagation were instantaneous, and we could adapt measurements to take advantage feedforward single-qubit gates only being applied half the time, a single layer of non-Clifford gates would still take a hard limit of the measurement time plus half the single qubit gate time.

Conclusion

We have investigated simple conditions that must be satisfied to realize a quantum advantage through polynomial speedups on a small fault-tolerant quantum computer. Our ultimate finding is that the prospects are generally poor for a quadratic speedup, consistent with folk knowledge in the error-correction community and recent work such as [7, 12]. The comparison to parallel classical resources is particularly damning for quantum computing and unfortunately, many quadratic quantum speedups (especially those leveraging amplitude amplification) apply to problems that are highly parallelizeable. The strongest conclusions in this work assume that one can achieve classical parallelism speedups on the order of 10^3 or more. But if one can produce a quadratic speedup for a problem where that is not the case, the prospects of quantum advantage would be improved.

These findings do not apply to all polynomial speedups. We found that while one would need to very significantly improving the rate of an error-corrected processor to help the case of quadratic speedups, having a quartic speedup rather than a quadratic speedup is often sufficient to restore the viability of achieving quantum advantage on a modest processor. Thus, we believe that these results suggest that the field should focus beyond quadratic speedups to find viable applications that might produce a quantum advantage on the first several generations of fault-tolerant quantum computers.

We expect this conclusion will persist under a variety of different cost models (e.g., were we to focus on the energy consumption of a computation rather than the runtime). However, we also expect that the community will make progress on some of the challenges described here, or perhaps identify circumstances under which the assumptions of this analysis do not apply. Either way, we hope that these arguments will foster further discussion about how we might develop broadly applicable algorithms that can achieve quantum advantage on small error-corrected quantum computers.

Acknowledgments

The authors thank Dave Bacon, Eddie Farhi, Austin Fowler, Bill Huggins, Sergei Isakov, Evan Jeffrey, Cody Jones, John Platt, Michael Newman, Rolando Somma, Nathan Wiebe and Will Zeng for helpful discussions and feedback on earlier drafts.

[1] R. P. Feynman, International Journal of Theoretical Physics **21**, 467 (1982).
[2] S. Lloyd, Science **273**, 1073 (1996).
[3] P. W. Shor, Proceedings 35th Annual Symposium on Foundations of Computer Science , 124 (1994).
[4] G. Brassard, P. Høyer, M. Mosca, and A. Tapp, in *Quantum Computation and Information*, edited by Vitaly I Voloshin, Samuel J. Lomonaco, and Howard E. Brandt (American Mathematical Society, Washington D.C., 2002) Chap. 3, pp. 53–74.
[5] L. K. Grover, in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96 (ACM, New York, NY, USA, 1996) pp. 212–219.
[6] R. D. Somma, S. Boixo, H. Barnum, and E. Knill, Physical Review Letters **101**, 130504 (2008).
[7] E. Campbell, A. Khurana, and A. Montanaro, (2018), 10.22331/q-2019-07-18-167.
[8] A. Montanaro, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **471**, 20150301 (2015).
[9] P. Rebentrost, B. Gupt, and T. R. Bromley, Physical Review A **98**, 022321 (2018).
[10] E. Aïmeur, G. Brassard, and S. Gambs, in *Advances in Artificial Intelligence*, edited by L. Lamontagne and M. Marchand (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006) pp. 431–442.
[11] N. Wiebe, A. Kapoor, and K. M. Svore, Quantum Info. Comput. **15**, 316–356 (2015).
[12] Y. R. Sanders, D. W. Berry, P. C. S. Costa, N. Wiebe, C. Gidney, H. Neven, and R. Babbush, arXiv:2007.07391 (2020).
[13] C. Gidney and A. G. Fowler, Quantum **3**, 135 (2019).
[14] I. D. Kivlichan, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, W. Sun, Z. Jiang, N. Rubin, A. Fowler, A. Aspuru-Guzik, H. Neven, and R. Babbush, Quantum **4**, 296 (2020).
[15] V. von Burg, G. H. Low, T. Haner, D. Steiger, M. Reiher, M. Roetteler, and M. Troyer, arXiv:2007.14460 (2020).
[16] J. Lee, D. Berry, C. Gidney, W. Huggins, J. McClean, N. Wiebe, and R. Babbush, arXiv:2011.03494 (2020).
[17] J. Lemieux, G. Duclos-Cianci, D. Sénéchal, and D. Poulin, (2020).
[18] A. M. Childs, D. Maslov, Y. Nam, N. J. Ross, and Y. Su, Proceedings of the National Academy of Sciences **115**, 9456 (2018).
[19] C. Gidney and M. Ekerå, (2019).
[20] A. Y. Kitaev, (1997), 10.1016/s0003-4916(02)00018-0.
[21] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, Physical Review A **86**, 32324 (2012).
[22] G. M. Amdahl, in *AFIPS '67 (Spring): Proceedings of the April 18-20, 1967, Spring Joint Computer Conference* (1967) pp. 483–485.

[23] J. L. Gustafson, Communications of the ACM **31**, 532 (1988).
[24] A. G. Fowler and C. Gidney, arXiv:1808.06709 (2018).
[25] C. Jones, Physical Review A **87**, 22328 (2013).
[26] B. Eastin, Physical Review A **87**, 032321 (2013).
[27] R. D. Somma, S. Boixo, H. Barnum, and E. Knill, Physical Review Letters **101**, 130504 (2008).
[28] J. Lemieux, B. Heim, D. Poulin, K. Svore, and M. Troyer, Quantum **4**, 287 (2020).
[29] S. Boixo, E. Knill, and R. Somma, Quantum Information & Computation **9**, 0833 (2009).
[30] S. Kirkpatrick, C. Gelatt Jr., and M. Vecchi, Science **220**, 671 (1983).
[31] S. V. Isakov, I. N. Zintchenko, T. F. Ronnow, and M. Troyer, Computer Physics Communications **192**, 265 (2015).
[32] A. Ambainis, K. Balodis, A. Belovs, T. Lee, M. Santha, and J. Smotrovs, arXiv:1506.04719 (2015).
[33] S. Aaronson, S. Ben-David, R. Kothari, and A. Tal, arXiv:2004.13231 (2020).
[34] A. W. Harrow, A. Hassidim, and S. Lloyd, Physical Review Letters **103**, 150502 (2009).
[35] C. Ballance, T. Harty, N. Linke, M. Sepiol, and D. Lucas, Physical Review Letters **117**, 060504 (2016).
[36] S. Crain, C. Cahall, G. Vrijsen, E. E. Wollman, M. D. Shaw, V. B. Verma, S. W. Nam, and J. Kim, Communications Physics **2**, 1 (2019).
[37] A. G. Fowler, arXiv:1210.4626 (2012).
[38] C. Gidney and A. G. Fowler, arXiv preprint arXiv:1905.08916 (2019).
[39] N. Delfosse, B. W. Reichardt, and K. M. Svore, arXiv preprint arXiv:2002.05180 (2020).
[40] B. Eastin and E. Knill, Physical Review Letters **102**, 110502 (2009).
[41] N. Cody Jones, J. D. Whitfield, P. L. McMahon, M.-H. Yung, R. V. Meter, A. Aspuru-Guzik, and Y. Yamamoto, New Journal of Physics **14**, 115023 (2012).
[42] A. G. Fowler, arXiv:1310.0863 (2013).

Appendix A: Accounting for error-correction costs

In the main text, we provide an estimate for the time that it takes to perform a single Toffoli gate with optimized factories within the surface code. The crux of the argument in the main text, is that this time is so much slower than the classical equivalent, there is a massive overhead which must be first overcome. We believe that it is valuable in directing future research in error-correction and algorithms to break down the origin of this overhead into its contributions from quantum error-correction and the physical device speed itself. Here we do so in some detail for the case of the surface code in superconducting qubits, and in passing for ion traps. We hope that this discussion will elucidate several avenues through which breakthroughs in error-correction might materially change the analysis of the main text.

To begin, we will assume that there is a physical two-qubit operation and syndrome measurement speed, τ and τ_s, where $\tau_s > \tau$ as τ is used to build measurement circuits along with a base physical measurement time τ_m. Modern fault-tolerant error-correction proceeds via rounds of syndrome extraction, processing, and correction in order to implement gates. The core physical operation of these rounds on the device is measurement of syndromes, and we are hence lower bounded by the measurement time τ_s in realistic settings. For context, estimates of these times for high fidelity superconducting qubits that would be realistic upon improvement are roughly $\tau \approx 10\,\text{ns}$ and $\tau_m \approx 100\,\text{ns}$. In comparison, for ion traps these numbers would be closer to $\tau \approx 100\,\mu\text{s}$ and $\tau_m \approx 10\,\mu\text{s}$ [35, 36]. Furthermore, at present, parallel operation is restricted in many-ion traps so a time complexity in the distance of the code is also required making a code cycle dramatically longer than this for large codes. For optimistic comparisons, we will assume that parallel operation at high fidelities has been achieved near the current gate rates of $100\,\mu\text{s}$ in the near future. If one had perfect operations, but still performed gates via a synthesized and fault-tolerant protocol, this represents the achievable runtime for a gate.

As our operations are not perfect, however, we will need to encode in an error-correcting code with some distance d which is chosen based on the error rate in our device, threshold of the code, and total number of operations we expect to perform. If one is allowed to use numerous ancilla qubits, this need not expand the runtime of individual operations by exploiting parallelism through teleportation and spacetime optimization [37, 38]. However, more qubit spartan implementations must use d rounds of measurement and correction to protect against measurement errors in the time direction, adding a factor of $\mathcal{O}(d)$ in the time cost. Research into one-shot correction techniques hopes to alleviate this time dependence on d without excessive space overhead [39], but current code constructions are not readily implementable.

On top of each round of these measurements, we must account for the time for this information to leave the device, be processed via decoding, and in some cases, implement active recovery after a gate, where this time depends on the hardware and complexity of the decoding. In order for error-correction to be efficient, it must be possible to process the syndrome data without an accumulation of rounds that grows in time. If we denote this processing latency as l_r, then the time for processing d rounds is lower bounded approximately by the time it takes to produce those syndrome measurements on the physical device plus this latency, or $(d\tau_s + l_r)$. Note that depending on the implementation details, l_r is likely to depend on d, but with sufficient classical parallelization it may be possible to make it effectively d independent.

On top of these costs, each gate has some associated prefactor in number of rounds that depends on the type of gate and its logical locality, C_G. For easy, or Clifford, gates in most codes, C_G can be made near 1. Unfortunately, in order to perform universal computation, one requires a gate which is not easy to implement [40], and common proposals center around state distillation where the prefactor C_G is often on the order of 10. Moreover, if

one considers synthesis of arbitrary rotations into multiple of these hard gates, C_G can multiply by a factor of 10 or more depending on the precision, leaving C_G on the order of 100. Putting these together, we can approximate a lower bound on the quantum gate time scaling in terms of error-correction parameters as

$$t_G \propto C_G \left(d\tau_s + l_r\right) \tag{A1}$$

Now that we have a general picture of how the time overhead enters for quantum error-correction, we examine it in a specific gate and context. In particular, we focus on superconducting qubits with feasible error rates and operation times within the surface code. Toffoli gates are required to implement classical logic on a quantum computer but cannot be implemented transversely within practical implementations of the surface code. Instead, one must implement these gates by first distilling resource states. To implement a Toffoli gate one requires a CCZ state ($|\mathrm{CCZ}\rangle = \mathrm{CCZ}\,|+++\rangle$) and these states are consumed during the implementation of the gate. Distilling CCZ states requires a substantial amount of both time and hardware and thus, they are usually the bottleneck in realizing quantum algorithms within the surface code.

Here, we will focus on the state-of-the-art Toffoli factory constructions of [13] which are based on applying the lattice surgery constructions of [24] to the fault-tolerant Toffoli protocols of [26, 41]. Using that approach one can distill one CCZ state using two levels of state distillation with $5.5\,d + \mathcal{O}(1)$ surface code cycles and a factory with a data qubit footprint of about $12\,d \times 6\,d$ where d is the code distance (the total footprint includes measurement qubits as well, and is thus roughly double this number). Hence for the Toffoli gate, we take $C_G \approx 5.5$.

We will assume a correlated-error minimum weight perfect matching decoder capable of keeping pace with 1 μs rounds of surface code error detection [42], and capable of performing with a similar latency of feedforward in about 1 μs for d around 30, and conservatively lower bound the overall time for d rounds to then be $(d\tau_s + l_r) \leq 30\,\mu s$. We will also assume physical gate error rates in the vicinity of 10^{-3}, which we hope will be achievable at scale in the next decade. Since we expect to require on the order of billions of Toffoli gates to achieve quantum advantage for practical applications (we will see this is actually a significant underestimate for the case of quadratic speedups) we will assume that a code distance in the vicinity of $d = 30$ will be sufficient (since errors are suppressed exponentially in code distance this number will be approximately correct).

With these assumptions, our model predicts a Toffoli gate time of $t_G = C_G(d\tau_s + l_r) \approx 5.5 \times 30\,\mu\mathrm{s} \approx 170\,\mu\mathrm{s}$. This rough approximation matches the more detailed resource estimate which shows the spacetime volume required to implement one Toffoli gate is approximately 23 qubit seconds [13]. We discuss the resources required for distillation in terms of qubitseconds because it is generally possible to make tradeoffs between space and time but the critical resource to minimize is actually the product of the two. Under these assumptions we would be able to distill a Toffoli gate in about 170 μs using around 130,000 physical qubits (see the resource estimation spreadsheet in [13] for detailed assumptions). Due to this large overhead we focus on estimates assuming we distill CCZ states in series, which is likely how we would operate early fault-tolerant surface code computers. For comparison, if ion trap implementations used a similar surface code implementation and error rates while dramatically improving syndrome measurement time to $\tau_s = 100\,\mu$s in parallel, the gate time assuming $C_G \approx 5.5$ is $t_G \approx 17{,}000\,\mu$s, or roughly a factor of 100 slower.

To make this more concrete, we can convert this to a unitless error-correction overhead for a particular gate of $C_G(d\tau_s + l_r)/\tau_s$. If we keep the $30\,\mu$s overall bound for $(d\tau_s + l_r)$, and make a reasonable estimate for the improvement of physical syndrome measurement times for superconducting qubits to 100 ns, then the error-correction overhead at this distance is 1,700.

This suggests that at present for superconducting qubits, the most fruitful improvements with regards to algorithmic speed are the reduction of decoding time, the minimization of time overheads in distillation factories, and then the reduction of number of measurement rounds required to protect in the time direction, perhaps through improved gate fidelities for equivalent operation times to result in lower required distances or through single shot protocols. If this can be achieved, the next milestones would be the reduction of physical syndrome extraction time. However, even such advances would already make prospects for realizing a quantum advantage with quadratic speedups considerably more enticing.