Quantum Supremacy Is Both Closer and Farther than It Appears

Igor L. Markov[1], Aneeqa Fatima[1], Sergei V. Isakov[2], and Sergio Boixo[3]

[1] University of Michigan, 2260 Hayward St, Ann Arbor, MI 48109
[2] Google Inc., 8002 Zürich, Switzerland
[3] Google Inc., Venice, CA, 90291

August 17, 2018

Abstract

As quantum computers improve in the number of qubits and fidelity, the question of when they surpass state-of-the-art classical computation for a well-defined computational task is attracting much attention. The leading candidate task for this *quantum computational supremacy* milestone entails sampling from the output distribution defined by a random quantum circuit. We perform this task on conventional computers for larger circuits than in previous results, by trading circuit fidelity for computational resources to match the fidelity of a given quantum computer. By using publicly available Google Cloud Computing, we can price such simulations and enable comparisons by total cost across multiple hardware types. We simulate approximate sampling from the output of a circuit with 7×8 qubits and depth $1 + 40 + 1$ by producing one million bitstring probabilities with fidelity 0.5%, at an estimated cost of $35184. The simulation costs scale linearly with fidelity, and using this scaling we estimate that extending circuit depth to $1 + 48 + 1$ increases costs to one million dollars. Yet, for a quantum computer, approximate sampling would take seconds. We pay particular attention to validating simulation results. Finally, we explain why recently refined benchmarks substantially increase computation cost of leading simulators, halving the circuit depth that can be simulated within the same time.

1 Introduction

The promise of quantum computers inspired the pursuit of *quantum computational supremacy*, i.e., using quantum computers to solve some task that is prohibitively hard for conventional computers [1, 2, 3, 4, 5, 6, 7, 8, 9]. A key benchmark has emerged in terms of randomized, universal quantum circuits proposed by Google [5], and this inspired rapid progress in algorithms for simulating quantum computers [10, 5, 11, 12, 13, 14, 15, 16, 17]. Simulating sampling or performing cross-entropy benchmarking [5] requires calculating a large number of output probabilities requested at random. We focus on a method that essentially evolves the quantum wave function and outputs a large number of probabilities with small additional cost [7, 14]. We show that two recent quantum simulations that required supercomputers with massive amounts of memory [11, 12] can now be performed with modest amounts of space and time using publicly accessible cloud computing. To match the accuracy of results in a given quantum computer with arbitrary quantum gates, we developed an approximate simulation whose runtime scales linearly with circuit fidelity. In contrast to the recent use of the world's most powerful supercomputer (over 100 Pflops, 131K nodes) to simulate Google circuits [15], we add a new dimension to quantum-versus-classical comparisons and report monetary cost of our simulations in terms of cloud-computing resources. This metric captures rapid progress in quantum-circuit simulation since 2016. On the other hand, we show that

- Small changes to common quantum-supremacy circuits make them considerably harder to simulate.
- The use of more sophisticated quantum gates substantially handicaps leading simulation methods.

In 2017 and 2018, IBM, Intel and Google announced new quantum computing chips that implement 50 and 72 qubits [18, 19, 20], but no empirical results for these chips have yet been announced as the chips are being improved. These chips will have to compete with simulation software running on conventional computers. The most straightforward approach — Schrödinger-style simulation that maintains and modifies the wave-function — requires 2^n-sized memory for n qubits [10]. Therefore, it is possible to simulate 30-qubit circuits on a laptop, and 35-qubit circuits on a mid-range server. Further scaling would seem to

require supercomputers, and a 2016 simulation [11] used 0.5 PB RAM on a Cori II supercomputer to simulate a depth-26 45-qubit circuit proposed by Google for quantum supremacy experiments [5]. These universal circuits were designed to run on planar $\sqrt{n} \times \sqrt{n}$ qubit-array architectures [5] and evade easy simulation. They start with a Hadamard (H) gate on every qubit and arrange nearest-neighbor controlled-Z (CZ) gates in a repeated pattern, so that every possible CZ gate appears once every eight cycles. One-qubit $X^{\frac{1}{2}}$, $Y^{\frac{1}{2}}$ and T gates are randomly interspersed between CZ gates so as to prevent cancellations and induce chaotic quantum dynamics [5], as illustrated in [11, Figure 1]. Each circuit ends with a measurement on every qubit. On a quantum computer this measurement produces an unpredictable bitstring and thus samples the distribution determined by the specific quantum circuit. Competing simulation techniques must calculate a large enough number of amplitudes (each corresponding to some bitstring) to simulate sampling from the output distribution [21, 5, 13].

Common sense suggests that circuits with more than 49 qubits would require too much memory to simulate in practice. To the contrary, alternative methods related to Feynman paths [22], tensor network contractions [23, 13], and similar approaches [7, 14] can trade space for time complexity. Google researchers have shown that for circuits with low depth, one can quickly find any one amplitude on a single computer for over 100 qubits using a variable elimination algorithm [13], and have illustrated the statistical distribution of the output of a circuit with 7×8 qubits and depth 30 by computing 2×10^5 probabilities using multiple computers. In April 2018, a simulation on the world's largest supercomputer, Sunway TaihuLight [15], computed 2^{46} amplitudes for a circuit with 7×7 qubits and depth $1 + 39$. Our notation for depth (introduced here) explicitly denotes layers of Hadamard gates with "1+" at the beginning of the circuit (clock cycle 0) and "+1" at the end. The remaining 39 layers (clock cycles) include CZ gates and other, one-qubit, gates. A related computation produced a single amplitude for a circuit of depth 55. The depth attained in these simulations includes 8 cycles gained by exploiting an unfortunate design choice — sequences of diagonal gates CZ - T - CZ, — at the beginning of the circuit, specific to the circuits in Ref. [5] and avoided in revised benchmarks [24] as explained in Section 5.

All simulations discussed so far seek exact output amplitudes, aside from negligible numerical errors. However, the computational task of interest is to *approximately* sample from the bitstring distribution defined by a random quantum circuit, and near-term quantum computers incur significant, unavoidable errors because gate errors accumulate exponentially [3]. In particular, for Google quantum supremacy circuits [5, 24] with 7×7 qubits and depth $1 + 40 + 1$, a reasonable goal is to achieve a 0.005 circuit fidelity, consistent with a two-qubit gate fidelity of 0.995, one-qubit gate fidelity of 0.999, initialization fidelity of 0.998 and measurement fidelity of 0.99 [25, 26, 27, 5] (in this example, depth $1 + 40 + 1$ entails initial and final Hadamard gates on each qubit). The same circuit fidelity at depth $1 + 48 + 1$ would require a two-qubit gate fidelity of 0.996. In our simulations, we configure circuit fidelity values $f = 0.01$ and $f = 0.005$, and show how to revise resource estimates to any given $f \in (0, 1]$, using linear scaling that we prove. Significantly, rather than simulate individual gates to some fixed fidelity, we control simulation fidelity for the entire circuit.

Extending recent progress in quantum circuit simulation algorithms, we implemented massively-parallel simulation without inter-process communication (IPC) and proprietary hardware that can run in commercial cloud-based *preemptible virtual machines* (VMs).[1] We report the following results:

- Using a single `n1-highcpu-96 server` from Google Cloud, we calculate one million probabilities (requested at random) per simulation, enough to simulate sampling [21, 13] for circuits used in two recent supercomputer simulations:
  - For a circuit with 6×7 qubits and depth $1 + 25$ [5], our simulation took 4.7 hours, with estimated cost $3.34 in the cloud. This is significantly cheaper and more accessible than simulations in Ref. [5], but we recall that those simulations study the convergence to the Porter-Thomas distribution as a function of depth with exponential precision.
  - For a circuit with 9×5 qubits and depth $1 + 25$ [11] our simulation took 20 minutes, with estimated cost $0.24 in the cloud. We use 17.4GiB peak memory where Ref. [11] used 0.5PB — a 28600 times improvement.[2] Notably, for methods that compute one amplitude at a time, 9×5 circuits are not substantially harder than 5×5 circuits [13].

These results are reported only to compare our simulation to prior work. We also calculate one million probabilities for the circuit `inst_7x6_26_0` [24] with 7×6 qubits and depth $1 + 25 + 1$. This refined benchmark is substantially harder to simulate for some methods (see Section 5). The runtime was 6 hours (as apposed to 4.7 hours for the previous circuit of similar size), corresponding to a cost of $4.32.

---

[1]A preemptible VM is cheaper than an on-demand VM because it does not guarantee real-time execution. A process may be terminated (preempted) by a higher-priority task at any time, but can then be restarted later. Pricing for preemptible VMs in a commercial cloud varies for different hardware configurations and may fluctuate hourly with demand for computational resources.

[2]Our peak memory usage can be improved or traded for runtime at the same cost point.

	Supercomputer simulations	No-IPC distributed simulations
Total memory and CPUs	Great, but for short time intervals	Modest, but available for longer time
Researchers' access to HW	Limited: must adapt to existing HW config and availability in time	Easy: cloud services and universities offer many HW configurations
Time-space tradeoffs	System- and algorithm-dependent	Straightforward (iso total core-hours)
Sys-specific programming	Extensive (NUMA, data transfers)	None
Memory buffers	Needed for data transfers [11, 12]	Not needed
SW reuse on diverse HW	Computational kernels reuse	Full SW reuse between systems
Job scheduling	Synchronization needed	No synchronization needed
Communication time-outs and job failures	May force a restart of the entire simulation	Easy to tolerate by restarting failed jobs only
Preemptible VMs in cloud	Incompatible	Straightforward to use
Scalability	Depends on communication	Linear, unaffected by communication
Performance estimation	System-specific	Straightforward
Simulation costs	Significant, hard to quantify	Relatively low, easy to forecast, but may fluctuate with cloud demand

Table 1: Comparing simulations of quantum circuits on (*i*) supercomputers [5, 11, 12, 15] that rely on fast interconnect between nodes to (*ii*) simulations on distributed clusters [14] with no interprocess communication (IPC). Comparing such simulations directly is difficult, but total core-hours may be a useful metric. Differences in the sophistication and performance of CPUs may be reflected in the total cost of simulation. Available memory and its speed may also affect total costs and may encourage different types of simulation algorithms for supercomputer simulations.

- We propose a new approach to approximate simulation of quantum circuits with arbitrary gate libraries. In particular, we compute one million probabilities with fidelity 0.5% for the recently proposed `inst_7x7_41_0` circuit [24] with 7×7 qubits, depth $1 + 40 + 1$ and controlled-Z (CZ) two qubit gates using 625 `n1-highcpu-32` servers from Google Cloud, with estimated cost $8734.

- We propose an interactive protocol to validate the results of quantum-supremacy simulations without knowing the correct answer. The protocol uses (*i*) approximate simulation with prescribed fidelity, and (*ii*) inner-product estimation.

In Section 5, we summarize weaknesses found in quantum supremacy circuits [5] simulated in prior work and introduce new benchmarks that are more difficult to simulate.

While contrasting our work on exact simulation with prior art, we anticipate that other algorithms may improve, but here we refer to published results. Our simulation produces a large number of requested amplitudes at once (cf. [13, 16]) and is not specialized to individual circuits (cf. [15, 14, 16]). It can be configured to run on a distributed cluster without interprocess communication (IPC) rather than on a tightly coupled supercomputer (cf. [11, 12, 15]) — see Table 1 for a comparison. Our use of modest computing hardware resources democratizes quantum circuit simulation. Recent work related to ours [14] used GPUs and a distributed cluster with 128 nodes to simulate quantum circuits. The layout of CZ two-qubit gates used in that work was changed relative to the prior work to lower the cost of classical simulations, which prevents a direct comparison of runtimes in general: we focus instead on the circuits from Ref. [24], which are significantly harder to simulate. Nevertheless, we performed a simulation of a circuit with 8×8 qubits and depth $1 + 22$ with the choice for the layout of CZs from Ref. [14]. Our simulation took 18.6 hours on 64 `n1-highmem-32` Google Cloud virtual machines, a $1.7\times$ runtime improvement using half as many nodes and running in a commercial cloud environment.[3]

2 Our quantum-circuit simulation framework

Our algorithms can be characterized as Schrödinger-Feynman hybrids [7, 14]. In the context of nearest-neighbor quantum architectures, we partition a given qubit layout into blocks. We then decompose quantum gates acting across the partition into sums of separable terms, such that, for each term, each block can be simulated independently and the results can be added up. For example, using a pair of half-sized qubit

---

[3]The nodes in the simulation of Ref. [14] have each a Xeon E5-2690V4 processor with 14 cores. We use `n1-highmem-32` virtual machines in Google Cloud, with 32 vCPUs or hyper-threads, equivalent to 16 cores 2.0 GHz Intel (Skylake) platform.

blocks instead of representing a full wave function reduces memory requirements for k qubits from 2^k to $2^{k/2+1}$, but introduces a dependency on the number of decomposed gates. For CZ gates the decomposition has two terms [4]

$$\text{CZ} = \text{diag}(1,1,1,-1) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = \text{P}_0 \otimes \text{I} + \text{P}_1 \otimes \text{Z} . \quad (1)$$

Thus, applying each *cross-block* CZ gate (xCZ) to a tensor term produces two tensor product terms, doubling runtime.[5] In comparison, traditional Feynman-style path summation [22, 5] uses very small amounts of memory, but doubles its runtime on every (branching) gate, resulting in much longer runtime and not being able to use available memory fully. Our simulator combines highly-optimized Schrödinger-style simulation within each qubit block and simulates xCZ gates with Feynman-style path summation, to limit memory use. Unlike in Feynman-style simulation, runtime scales with the number of xCZ gates, which is very limited in planar qubit-array architectures with nearest-neighbor gates [7]. Unlike traditional Schrödinger-style simulation, the resulting algorithms are depth-limited, and supercomputer simulations may hold some advantage for very deep circuits. However, near-term quantum computers rely on noisy gates [3] that also limit circuit depth.

Our Schrödinger-style simulation includes optimizations that help with arbitrary gate libraries and some that help with small gate libraries, while others target the Clifford+T and related gate libraries [6] These optimizations equally apply when using supercomputers, laptops and other hardware.

- Clustering gates of a kind (with reordering), rather than gates acting on the same qubits as in prior simulations. For example, in Google quantum supremacy circuits [5, 24], we collect separate clusters of CZ, T, $X^{\frac{1}{2}}$ and $Y^{\frac{1}{2}}$ gates.[7]
- Compact encoding of gate positions in each cluster into bitmasks, so that efficient bitwise operations (parity, population count, as well as counts of trailing and leading zero bits) and mod-8 arithmetics (for T gates) can apply all diagonal gates in a single pass over the wave-function or its slices.[8]
- Cache-efficient algorithms to simulate large gate clusters. Instead of accessing an entire wave-function, we apply gate clusters on amplitude slices sized to fit in the L2 cache.[9]

Compared to the state of the art described in [11], we can eliminate floating-point multiplications when simulating Google circuits (and only use them for convenience when simulating T gates). The number of floating-point additions is reduced by a factor of three, while fully benefiting from vectorized instruction extensions (AVX2) in modern CPUs. This puts emphasis on cache performance and limits thread scalability. By streamlining memory accesses, we improve single-thread performance as well as scaling to multiple threads and multiple processes using the same memory bank. However, the impact of subsequent improvements is even greater.

For Feynman-style path summation, our simulator optimizes performance in several ways. When simulating a circuit with $x > 0$ xCZ gates, we use the technique suggested in [7] and demonstrated in [14] where each process receives a bitstring that encodes a unique path, then save the simulation results in a file for subsequent summation.[10] This is convenient because no interprocess communication is required, but many processes simulate the same path prefixes. We therefore subdivide $x = x_p + x_b$ and use only x_p-bit prefixes to spawn separate processes. After the first x_p bits, each process checkpoints its simulation state (doubling its memory usage once) and simulates each of 2^{x_b} remaining paths starting from this checkpoint. A second optimization for Feynman-style path summation is supported by the Schrödinger-style simulator algorithm: we noticed that the use of projection operators in gate decompositions such as Equation (1) leads to a large number of zeros in block-local wave-functions. Therefore, our simulator skips blocks of zero amplitudes rather than apply quantum gates to them.

Final-state amplitudes are calculated for any requested subset of indices, unlike simulators in [12, 15, 16].

---

[4]A two-qubit gate can be decomposed into at most four tensor-product terms, e.g., with one-side operator basis P_0, P_1, $|0\rangle\langle1|$, $|1\rangle\langle0|$.

[5]Moreover, new pairs of tensor terms are orthogonal because their first components (produced by P_0 and P_1) are. While unitary operators preserve their orthogonality, projections can increase their cosine similarity.

[6]Clifford+T and related gate libraries are common because they promise compatibility with quantum error-correction, they are used in Google and IBM quantum computers.

[7]This technique might not help in circuits that use a large number of different gate types.

[8]Some of these techniques exploit specific gate types, and our simulator may run $2-3\times$ slower on a different gate library.

[9]This technique does not depend on specific gate types.

[10]Our simulator was implemented before [14] was described publicly.

This is necessary to simulate sampling [21] and to perform cross-entropy benchmarking [5]. Subsets of millions of amplitudes are calculated almost as quickly as single amplitudes (c.f. [13]). However, if a very large subset is requested, such calculations may dominate the cost of "easy" simulations. Contributions to requested amplitudes are accumulated over simulation paths. When used in its *exact mode*, the simulator calculates each amplitude with a negligibly small numerical error.[11]

3   Trading off simulation fidelity for computational resources

We are interested in performing the following computational task proposed for a near-term quantum supremacy demonstration: approximately sampling from the output distribution defined by a random quantum circuit [5, 7]. Quantum computers experience errors in qubit initialization, gates and measurements. This suggests speeding up the simulation by reducing the accuracy of results to the level attained by quantum computers. We are not aware of such attempts for simulating quantum supremacy circuits, but the work in [28, **?**] developed approximate simulation of circuits dominated by Clifford gates.

Our approach to approximate simulation can handle arbitrary quantum gates, and our fidelity scaling is very different from error scaling in Refs. [28, **?**], as shown below. Equation (1) shows that each xCZ results in two tensor-product terms (up to four terms for other gates). Simulating x such xCZ gates (among other gates) produces 2^x simulation paths, corresponding to all possible choices of either $P_0 \otimes I$ or $P_1 \otimes Z$ for each xCZ, see Equation (1) . Therefore we can write the output of the simulation as

$$|\psi\rangle = \sum_{j=0}^{2^x-1} |\varphi_j\rangle \,, \tag{2}$$

where j enumerates the different paths. To approximate this sum, one can drop some of the terms. In the context of multiprocess simulation, one can simply skip some of the processes. When simulating quantum-supremacy circuits [5], the 2^x terms tend to have nearly-identical norms because odd amplitudes and even amplitudes selected by projections in Equation (1) are equally distributed. This is the worst case (otherwise, we would have dropped terms with lower norms), so we assume $\||\varphi_j\rangle\| = 2^{-x/2} \; \forall j$. Furthermore, each path corresponds to a different quantum random circuit, and therefore their output states are almost orthogonal, $|\langle \varphi_j | \varphi_k \rangle|^2 \simeq 2^{-x \cdot n}$ for $j \neq k$. For requested simulation fidelity $0 < f \leq 1$, we include $f2^x$ terms out of 2^x

$$|\psi_a\rangle = \sum_{j=0}^{f2^x-1} |\varphi_j\rangle, \;\; \text{so} \;\; \langle \psi_a | \psi_a \rangle = f2^{-x} \text{(see Figure 2a)} \,. \tag{3}$$

Fidelity can then be estimated as

$$\frac{\langle \psi | \psi_a \rangle \langle \psi_a | \psi \rangle}{\langle \psi_a | \psi_a \rangle} = \frac{f^2 2^{-x}}{f2^{-x}} = f \,, \tag{4}$$

see Figure 2b. Attaining fidelity f with only an f fraction of work is remarkable, e.g., quantum simulation with fidelity 0.1 is 10× faster than exact simulation. This scaling holds up well in multiprocess simulation, as we confirmed empirically. The specific selection as to which $|\varphi_j\rangle$ terms to leave in Equation (3) does not matter for Google quantum supremacy circuits because the norms $\||\varphi_j\rangle\|$ are nearly identical (we checked this). Nevertheless, in general, eliminating terms with smaller norms can deliver a better-than-linear tradeoff between fidelity and runtime.

Approximate simulation methods in [28, **?**] do not appear particularly promising on quantum-supremacy circuits [5], which have numerous T gates (see Tables 2 and 3). The results in [28, **?**] differ from ours in another essential way. Their runtime scales with $1/\epsilon^2$ for circuit error-rate $\epsilon > 0$, which is beneficial for *small-error simulation*. Our scaling, linear in f, is beneficial for *small-fidelity simulation* and near-term quantum-supremacy experiments.

Any cross-block multiqubit gate can be handled with our techniques by replacing the decomposition in Equation (1) by an operator Schmidt decomposition of the gate, ensuring the minimal number of terms. The linear scaling of runtime with fidelity does not depend on this Schmidt rank.

Validation of simulation results for Google quantum-supremacy circuits is facilitated by approximate simulation. Consider the correct final state $|\psi\rangle$ of simulation and a norm-1 approximate state $|\psi_a\rangle$ with

---

[11]As a check, we calculate the norm of the state vector using higher-precision accumulator variables and underflow mitigation. Comparing the result to 1.0, we observe a very close match in all cases when such a match is expected.

$|\langle\psi|\psi_a\rangle|^2 = f$. As shown above, multiple $|\psi_a\rangle$ states can be produced f times faster (each) than the $|\psi\rangle$ state. Such approximate states help validate claims of producing $|\psi\rangle$. In the following two-party validation protocol, we assume a particular quantum circuit that the Verifier can simulate to fidelity $f < 1$ in the sense that the Verifier can approximate the final simulation state with fidelity f and do so with a sufficient variety of states at random. For each state, the Verifier and the Claimant save k amplitudes so as to estimate fidelities between their states to accuracy $\delta \leq f$.

1. The Verifier gives the Claimant k pseudo-randomly generated amplitude indices (we used k=1M and specified the indices by a PRNG seed).

2. The Claimant simulates the circuit and saves k amplitudes with given indices.

3. The Verifier picks a fidelity value $0 \ll f_1 < 1$ and simulates the circuit with fidelity f_1 producing one of many possible $|\psi_a\rangle$ states and saving k amplitudes with the specified indices.

4. The Claimant shares their amplitudes with the Verifier.

5. The Verifier estimates fidelity f_e between the Verifier's and the Claimant's states.

6. If $|f_e - f_1| > \delta$, the test fails. Else, the test may be repeated to increase the certainty of passing.

A handful of sufficiently unpredictable approximate states suffice to validate $|\psi\rangle$ because for a generic approximate state with fidelity $f_1 \ll 1$, the expected state fidelity w.r.t $|\psi_a\rangle$ is $f_1 f \ll f$. Moreover, the Verifier can estimate the fidelity of the Claimant's state. The Verifier can select circuits from some family and repeat the test for different circuits.

Approximate simulation in a cloud-computing environment is discussed in Section 4.

4 Performance estimation and simulation results

Performance estimation for our simulation is based on the following parameters. The qubit array is partitioned in two blocks $q = q_1 + q_2$, attempting to minimize the larger block and reduce the number of xCZ gates (x). We use a single straight line cut in the qubit array for each simulation. As explained before, we split $x = x_p + x_b$. The corresponding split for circuit depth is $d = d_p + d_b$. The simulation is configured to save $n_a > 0$ amplitudes with specified indices. The total runtime of all simulation processes (however many threads each process may use) can then be estimated as

$$T_{tot}(f, q_1, q_2, d_p, d_b, x_p, x_b, n_a) = C_1 f 2^{x_p}(q_1 2^{q_1} + q_2 2^{q_2})(d_p + C_2 2^{x_b} d_b) + C_3 2^{x_p + x_b} n_a \qquad (5)$$

where $C_1, C_2, C_3 > 0$ are implementation-specific constants. In each of the $d = d_p + d_b$ cycles, we simulate on the order of q gates, of which q_1 gates act on 2^{q_1} amplitudes and q_2 gates act on 2^{q_2} amplitudes. After the first d_p cycles, the simulation state is saved and reused in 2^{x_b} branches over d_b cycles. At the end of each branch, n_a requested amplitudes are collected (unlike the work in [12], our simulator can save any subset of amplitudes specified by indices). For $n_a = 10000$, the last term is significant for $q \leq 36$, but can be neglected when $n_a \ll 2^{q_1} + 2^{q_2}$. Given a simulation cluster that can run p simulation processes per node on N nodes, we distinguish billable time and wallclock time

$$T_{\text{bill}} = \omega(p) T_{\text{tot}}/p \qquad T_{\text{clock}} = T_{\text{bill}}/N \qquad (6)$$

where $\omega(p) \geq 1$ reflects the slowdown of an individual process sharing a node with other $p-1$ such processes, e.g., due to memory contention. While $\omega(1) = 1$, for the more common case $16 \leq p \leq 32$, $\omega(p)$ can be in the 1.5-2 range, depending on how much memory bandwidth a particular system offers and how much bandwidth is required by each process. In terms of RAM usage, a single process requires

$$M_{\text{proc}} = (C_4(2^{q_1} + 2^{q_2}) + n_a) \cdot \texttt{sizeof(complex)} \qquad (7)$$

bytes, where $C_4 > 0$ is a small integer. Each compute node requires $p M_{\text{proc}}$ bytes, and the entire cluster peaks at $p N M_{\text{proc}}$ bytes. Circuit depth does not affect RAM requirements.

For the simulations reported below, the billable time and wallclock time are measured directly.

Infrastructure used in our simulations has been as a follows. Key algorithms are implemented in C++11 in a package called Rollright (University of Michigan) and compiled with g++ 7.2. Our kernels use AVX-2 instructions, but not AVX-512 instructions used in [11]. We also use python scripts for several tasks. Empirical performance is evaluated on quantum supremacy circuits [5, 24], and results are shown in Tables 2 and 3, with additional details given in the appendices. Unlike prior published efforts, we use no GPUs, no supercomputers, and no high-performance node-to-node interconnect. Software development was performed on a MacBook Pro 2017 with 16 GiB RAM, where our baseline Schrödinger simulation completes

| Qubit | Circuit | Gate counts | | | | RAM | Runtime | Cost |
array	depth	total	T	CZ	xCZ	GiB	hr	$
6×7	$1 + 25$	600	131	222	18	8.58	4.6	3.34
7×6	$1 + 25 + 1$	711	157	224	18	8.58	6.0	4.32
9×5	$1 + 25$	643	139	238	15	17.4	0.3	0.24
9×5	$1 + 25 + 1$	767	176	237	15	17.4	1.4	1.01

Table 2: Exact simulation of "easy" quantum-supremacy circuits used in [5, 11] on a single Google Cloud `n1-highcpu-96` server with 96 hyper-threads, equivalent to 48 cores of 2.0 GHz Intel (Skylake) CPUs. Circuit depth is given in cycles, where the initial and final 1s represents Hadamard gates. Each simulation saved 1M amplitudes, requested at random (the I/O overhead is included in runtime). Cost estimates are based on $0.72/hr pricing for preemptible VMs. Circuits with 7×6 and 9×5 qubits and depth $1 + 25 + 1$ represent refined benchmarks from Section 5, including the circuit file `inst_7x6_26_0` [24].

a 30-qubit circuit with depth $1 + 26 + 1$ in 71 s using a little over 8 GiB of RAM. For circuits with more than 35 qubits, we use the Google Cloud, a common cloud-computing service where users rent networked servers of various types. We used several virtual server types (specified for each simulation below) with Ubuntu Linux, available to the public for fair comparisons. Most of the RAM available on those servers was not used in our simulations. For each server type, we report hourly rental cost in preemptible mode as of June 2018. Cost comparisons can be performed with other server types, adjusted for other price points, and extended to include quantum computers, so as to establish or refute quantum supremacy.

For simulations of "easy" quantum circuits we use 4 threads per process with 22 parallel processes on one `n1-highcpu-96` server with 96 vCPUs. [12] Simulations for up to 45 qubits were performed in exact mode, in configurations that match those previously reported, so as to facilitate comparisons (the gate counts do not always match exactly, but these differences have only minor impact on results). As implied by Table 2, we use much more modest resources than supercomputer-based simulations from 2016 and 2017, and achieve lower runtimes than cluster-based simulations. In particular, a 45-qubit simulation of depth $1 + 25$, previously performed on a supercomputer with 0.5PB RAM [11], took $0.24 and 17.4 GiB RAM with our algorithms. This illustrates rapid progress in common understanding and availability of quantum circuit simulation. To put our results in perspective, Appendix A compares Rollright to the most recent version of the Microsoft QDK simulator.

Practical validation of results is a critically-important step in quantum-supremacy simulations because subtle bugs or misguided optimizations can significantly improve runtime, while producing incorrect results. For circuits of up to 32 qubits we validated our main Schrödinger simulator against several independently-developed simulators based on different algorithms, to ensure that the output amplitudes match with good accuracy. Of these simulators, the publicly available QuIDDPro [29] could not simulate quantum supremacy circuits beyond 20 qubits. We then simulated circuits using our Schrödinger-Feynman simulator and validated results against a Schrödinger-only simulation, and against a variable elimination simulation [13] on low-depth circuits with up to 64 qubits. Computing state norms offers a good sanity-check for both exact and approximate simulation. We also validated approximate simulation against exact simulation for low-depth circuits by producing amplitudes for the same subset of indices and estimating state-overlap (fidelity) based on these amplitudes. [13] Another sanity check is to plot the distribution of probability values for a given set of amplitudes. For exact simulation, the results should closely match the Porter-Thomas distribution, see Figure 1a. While this is not required for approximate simulation, our method also produces a Porter-Thomas distribution in this case, see Figure 1b.

To validate the results of approximate simulation in this work, we performed an exact simulation of circuit `inst_6x6_41_0.txt` (from [24]) with 6×6 qubits and depth $1 + 40 + 1$, and saved 10^5 amplitudes. We calculated the same amplitudes using 40 different random fractions for each $f = 1/16$, $1/32$, $1/64$, and $1/128$ of paths. Figure 2b shows that empirical fidelity matches the fraction of paths used. The runtimes scale linearly with f. We also performed an exact 7×7 qubit depth 17 simulation and saved 1M random amplitudes. Then we performed approximate simulation with requested fidelities $f = 10^{-4}$, $f = 10^{-2}$ and $f = 1/64$ and saved the same amplitudes. We estimated the fidelity of approximate state by computing the

---

[12]On Google Cloud, a virtual CPU is implemented as a single hardware hyper-thread.
[13]We also calculated cross-entropy difference [5]. This gave similar fidelity estimates, although estimation errors exhibited a greater variance.
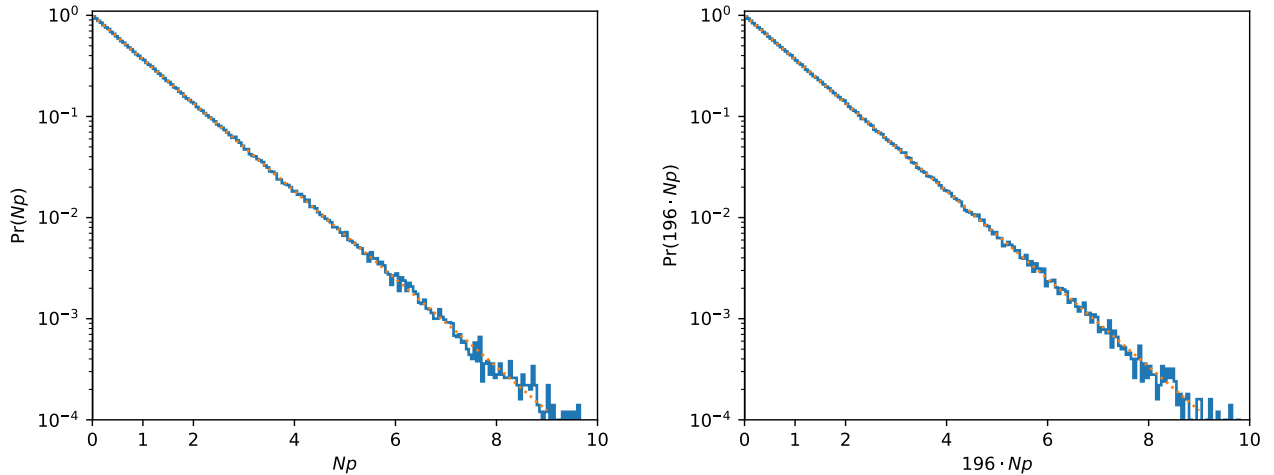
(a) Match to the Porter-Thomas distribution, 45 qubits.

(b) Porter-Thomas for 1/196 approximation, 56 qubits.

Figure 1: Distributions of simulated bitstring probabilities for the outputs of two random circuits. (a) Exact simulation for a circuit with 9×5 qubits and depth $1 + 25$: bitstring probabilities match the exponential (Porter-Thomas distribution) shown with a dashed line. (b) Approximate simulation with fidelity 1/196 for a circuit with 7×8 qubits and depth $1 + 40 + 1$, circuit file `inst_7x8_41_0` [24]. Bitstring probabilities again match the Porter-Thomas distribution, despite rather loose approximation.

inner product, and the fidelities matched the requested values to within 2%.

Runtime variation is important to track for two reasons.

- Simulators in some prior work [16] appear exponentially sensitive to quantum-supremacy circuit instances, possibly because they exploit easy-to-simulate gate sequences that can be removed (see Section 5). This is not the case in our work. For different circuits with the same depth and numbers of qubits, same placement of xCZs and similar number of gates, the runtime of our simulator is also
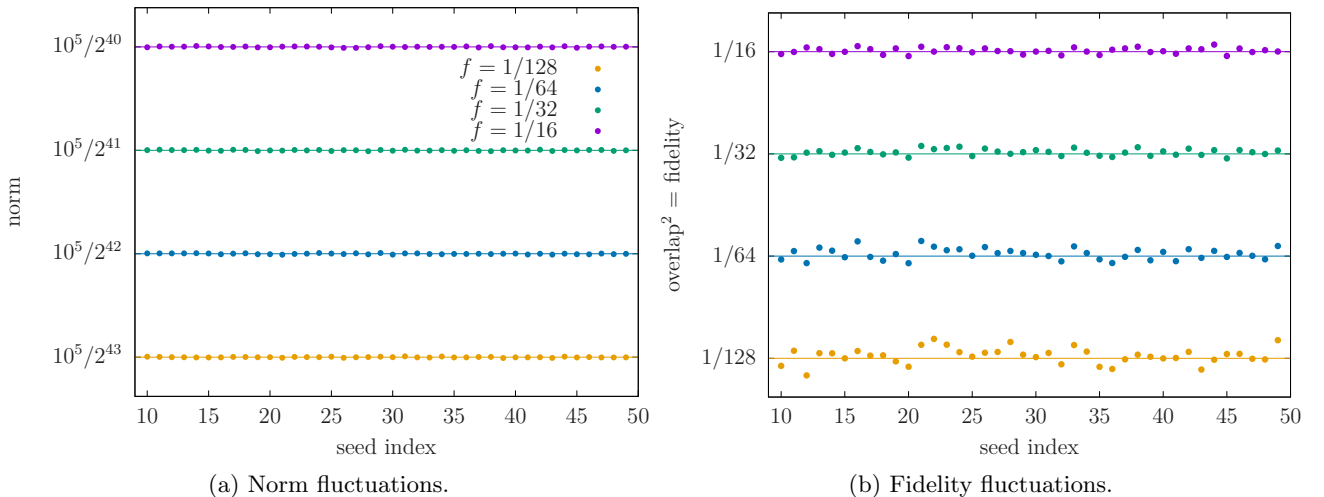
(a) Norm fluctuations.

(b) Fidelity fluctuations.

Figure 2: Fluctuations of results when simulating the 6×6-qubit circuit `inst_6x6_41_0.txt` (from [24]) with depth $1 + 40 + 1$. In each case we choose a random fraction $f =$ fidelity of xCZ paths to obtain state $|\psi_a\rangle$ from Equation 3, then estimate its norm and fidelity using 10^5 amplitudes. (a) The sum of 10^5 of the 2^{36} squared amplitudes of $|\psi_a\rangle$ is $10^5/2^{36} \cdot f$, as expected. (b) The fidelity (overlap^2) between $|\psi_a\rangle / \||\psi_a\rangle\|$ and the correct output is equal to the fraction f of xCZs paths used.

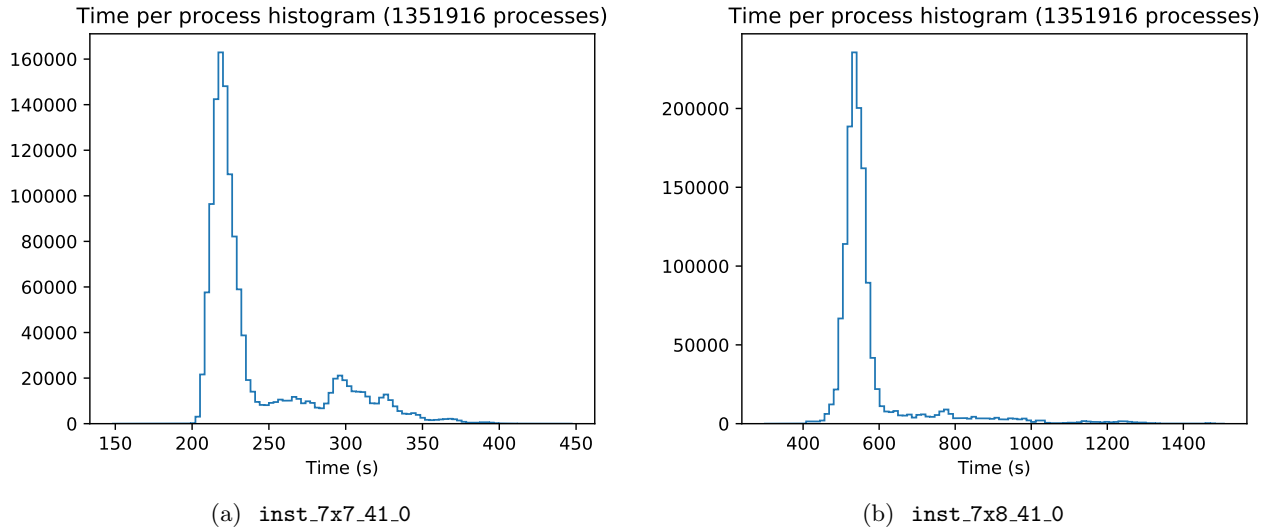|	
(a) `inst_7x7_41_0`	(b) `inst_7x8_41_0`

Figure 3: Single-process runtimes when simulating (*a*) a 7×7-qubit circuit with depth 1+40+1, and (*b*) a 7×8-qubit circuit with depth $1 + 40 + 1$. Hardware-performance variations are possibly due to CPU diversity and transient loads at cloud compute nodes. Process runtimes on the same node tend to be very similar. Each process computes 2^7 branches, corresponding to $x_b = 7$ in Eq. (5), for 10^6 amplitudes using 8 threads.

    similar. For a given circuit, the amount of work performed by individual branches of simulation and their runtimes on a given CPU do not exceed mean values for that CPU by more than 10-20%.

- Using servers, especially in a shared Cloud Computing environment, exposes simulations to hardware-performance variations possibly due to CPU diversity and transient loads at compute nodes (jobs by other users can schedule on the same compute nodes), see Figure 3.

Approximate simulation of "hard" quantum circuits. Simulations for 7×7 and 7×8 qubit arrays are more difficult than simulations we used for benchmarking purposes above, and are performed to a greater circuit depth in our work, $1 + 39 + 1$ and $1 + 40 + 1$, including the initial and the final layers of Hadamard gates [24]. Depth of at least 40 (plus Hadamards) was suggested [5, 13] as sufficient to be hard for simulations optimized for low depth circuits. It is easier to simulate circuits on oblong arrays than on square arrays, although specific algorithms may be affected in different ways [13]. For our simulation,

- we choose a cut with 7 xCZ gates every 8 cycles,

- the 7×7 array does not admit a balanced cut, and the blocks formed by the most balanced, smallest cut contain 28 and 21 qubits, whereas the 7×8 array admits an even $28 + 28$ partition.

This level of simulation difficulty offers a good opportunity to evaluate our approximate simulation technique. Therefore, we use $f = 10^{-2}$ for the 7×7 array depth $1 + 39 + 1$, and $f = 1/196$ for depth $1 + 40 + 1$. Distributed simulations were performed on shared physical hardware in Google Cloud, using up to 625 `n1-highcpu-32` virtual servers for 7×7 qubits and 625 `n1-highmem-32` servers for 7×8 qubits, with 32 vCPUs each. Key results are reported in Table 3, and details are available in appendices. These results can be used to estimate resources for different fidelity values f because runtime scales linearly with f. The linear scaling does not depend on the number of qubits or circuit depth.

    We also performed a simulation of a circuit with 7×7 qubits and depth $1+48+1$ with fidelity 2^{-22} using 512 `n1-highcpu-32` virtual servers with an estimated cost \$52. Using the linear scaling of computational cost with fidelity, this implies a cost of one million dollars for 0.5% fidelity.

5   Refined quantum-supremacy benchmarks

We now review the difficulty of Quantum Supremacy benchmarks for sampling the output distribution of random quantum circuits. Analysis in [5] suggests circuits with at least 7×7 qubits, as the quantum state of such a system is too large for typical Schrödinger-style simulations. With the same 7×7 array, circuit depth must be at least $1 + 40 + 1$, given the exponential growth of simulation difficulty with depth for simulation by variable-elimination [5, 13]. Note that the simulation of sampling requires calculating a large number

| Qubit | Circuit | Gate counts | | | | Fide- | RAM | Runtime, hr | | Cost |
array	depth	total	T	CZ	xCZ	lity	TiB	clock	billable	$
7×7	$1 + 39 + 1$	1252	286	410	31	1%	15	35.2	2.2e+04	5218
7×7	$1 + 40 + 1$	1280	293	420	35	0.51%	15	58.2	3.6e+04	8734
7×8	$1 + 40 + 1$	1466	331	485	35	0.51%	30	140.7	8.8e+04	35184

Table 3: Approximate simulation of "hard" quantum-supremacy circuits [24] on Google Cloud servers. Here we use version 2 benchmarks outlined in Section 5 that address weaknesses found in version 1 benchmarks [5]. The circuits are publicly available through [24]. Circuit depth is given in cycles, including the initial and final Hadamard gates on each qubit. Each simulation saved 1M amplitudes requested at random. The first simulation corresponds to circuit file `inst_7x7_40_0` and used 617 `n1-highcpu-32` machines, with cost estimate based on $0.24/hr pricing for preemptible VMs. The second simulation corresponds to circuit file `inst_7x7_41_0` and used 625 `n1-highcpu-32` machines. The last simulation is for circuit file `inst_7x8_41_0` and used 625 `n1-highmem-32` machines with cost estimates based on $0.40/hr pricing for preemptible processes.

of amplitudes requested at random [21], whereas variable elimination tends to produce one amplitude at a time. We have shown in previous sections how to simulate 7×7 circuits with CZ gates of depth $1 + 40 + 1$ trading computational cost for fidelity. This suggests revising the initially proposed benchmarks [5].

Quantum Supremacy entails an inherently adversarial protocol that asymmetrically favors quantum computers — a computational problem is being selected that can be solved by quantum yet not classical computers [1, 2, 3, 4, 5, 6, 7, 8, 9]. Reliably defeating Quantum Supremacy requires more than a handful of opportunistic simulations as one has to anticipate modifications of problem instances that complicate simulation more than they complicate quantum evolution. It is sometimes easy to confuse Quantum Supremacy with almost the opposite, i.e., showing that classical computers can solve tasks that present-day quantum computers cannot. The later is trivial and can be demonstrated on many common tasks for which classical software and hardware excel, while quantum computers have no algorithmic advantage and are currently much smaller and error-prone. For example, sorting 2^n numbers requires $\Omega(n2^n)$ time on both classical and quantum computers [30] and can be accomplished just as quickly. Of course, such statements do not preclude Quantum Supremacy. On the other hand, being able to quickly simulate some circuit benchmarks of a kind [31, 28, ?] but not others would not negate Quantum Supremacy. For instance, the work in [16] attempts to simulate a number of similar circuits but then abandons the ones for which simulation times out. The authors of [14] change the layout of CZ gates to make the simulation approximately 16 times faster at depth $1 + 22$. Simulations in [15, 16] attain additional depth by omitting the final layer of Hadamard gates, which makes the CZs in preceding cycles unnecessary. Moreover, the work in [16] calculates a single amplitude at a time, which is insufficient for sampling. Finding unexpected simulation shortcuts in existing quantum-supremacy benchmarks [5] is also of limited value if these shortcuts can be invalidated. To this end, the work in [15] describes how to simulate eight additional cycles in some cases by exploiting sequences of diagonal gates `CZ - T - CZ`. Avoiding such "easy" sequences during benchmark generation does not affect the execution on a quantum computer. Thus, Google benchmarks have been revised to

- replace every T gate appearing after a CZ gate with a non-diagonal gate,
- explicitly include a cycle of Hadamard gates before measurement,
- reorder cycles so that "horizontal" two-qubit gates alternate with "vertical" two-qubit gates.

While keeping the same gate family, these minimal changes complicate many approaches to simulation. As seen in Tables 2 and 3, the benchmarks retain numerous non-Clifford gates to hamper stabilizer-based simulation [28, ?]. Revised circuit benchmarks are publicly available [24] to facilitate fair runtime comparisons among different simulation methods. To ensure the replicability of results, we recommend mentioning specific circuit files, as we do in our work. The performance of our simulator on revised benchmarks is comparable to its performance on older benchmarks (see Table 2). The slowdown is moderate and primarily due to the slight increase in the number of gates, especially the Hadamards at the end of the circuit.

6 Harder benchmarks through two-qubit gates

In considering more drastic changes to circuit benchmarks, we estimate the impact of upgrading the gate library on simulation difficulty and on specific algorithms used in our work. To this end, replacing exist-

ing quantum gates within tensor blocks might require additional provisions for simulators that implement speedups for particular gate types, but would result only in a constant-times runtime difference. However, replacing cross-block CZ (xCZ) gates with more complicated gates can increase the branching factor in our multiprocessor simulation. The 4x4 matrix of a generic two-qubit gate can be decomposed into a sum of four tensor products of one-qubit gates

$$\left(\begin{array}{cc} A_{00} & A_{01} \\ A_{10} & A_{11} \end{array} \right) = \mathrm{P}_0 \otimes A_{00} + \mathrm{P}_1 \otimes A_{11} + |0\rangle\langle1| \otimes A_{01} + |1\rangle\langle0| \otimes A_{10} \tag{8}$$

where the A terms represent 2x2 blocks of A.[14] For example,

$$\mathrm{iSWAP} = \mathrm{P}_0 \otimes \mathrm{P}_0 + \mathrm{P}_1 \otimes \mathrm{P}_1 + i|0\rangle\langle1| \otimes |1\rangle\langle0| + i|1\rangle\langle0| \otimes |0\rangle\langle1| \ . \tag{9}$$

Note that non-unitary gates pose no difficulties to common simulation algorithms, and projections are particularly convenient because they create numerous intermediate zero amplitudes that can be skipped by simulators that track patterns of nonzeros (cf. Equation (10)). Lemma 1 in [32] shows that such decompositions cannot be shorter than the operator Schmidt decomposition (which must have rank 1, 2 or 4 for a unitary operator). The operator Schmidt decomposition for iSWAP

$$2 \cdot \mathrm{iSWAP} = \mathrm{I} \otimes \mathrm{I} + i\mathrm{X} \otimes \mathrm{X} + +i\mathrm{Y} \otimes \mathrm{Y} + \mathrm{Z} \otimes \mathrm{Z} \tag{10}$$

proves that the decomposition in Equation (9) is minimal. If all CZ gates are replaced with iSWAPs, the branching factor in multiprocess simulation increases from two to four, halving the circuit depth that can be handled within a given amount of time. The same is true of algorithms which map the circuit to undirected graphical models and use variable elimination [13, 16], as the corresponding treewidth grows roughly twice as fast.

Using an arbitrary two-qubit gate can slow down simulation of individual qubit blocks and also decrease the number of intermediate zero amplitudes, but the iSWAP gate has been well-studied and implemented in leading quantum chips [33, 34]. Moreover, an arbitrary two-qubit gate may be easier to approximate than an iSWAP by truncating the least-norm terms in its operator Schmidt decomposition. To create a new benchmark suite, while preserving the structure of existing benchmarks, we replace every CZ gate with an iSWAP gate. This benchmark suite is also available at [24].

In addition to considering more general one-qubit and two-qubit gates, the benchmarks can be modified to include two-qubit gates that couple arbitrary (non-adjacent) qubits. When such gates are applied within tensor partitions, our simulator will treat them just as efficiently as it treats gates in existing benchmarks. However, when such gates cross partitions, they will contribute to the branching factor of simulation. Nevertheless, our work focuses on simulating realistic quantum-circuit architectures, such as those pursued by IBM, Google, Rigetti and others. To this end, existing benchmarks are sufficient.

7 Conclusions

Universal quantum computers based on several different technologies started scaling beyond a handful of qubits around 2011. Today, detailed reports are published of programmable quantum-circuit computers with 22 qubits, while 50-qubit systems and larger are under testing [18, 19, 20]. Two-qubit gate fidelities have improved within the 0.9 - 0.995 range [25, 26, 27], but continue to limit implementable circuit depth. Numerical simulations traditionally focused on straightforward full-wave-function kernels, where progress from 42 qubits in 2010 to 45 qubits in 2018 [17] had been bounded by exponential memory scaling. The more recent focus on limited-depth quantum circuits extended supercomputer-based simulations well past 50 qubits [12, 13, 14, 15], facilitated the use of GPUs and distributed cluster without interprocess communication [14], and enabled meaningful simulations on single servers [13, 14] for up to 100 qubits.

We show that full-circuit fidelity in quantum-circuit simulation can be traded off for computational resources, and a simulation can match requested circuit fidelity to compete with a given quantum computer. We also show that (i) competitive simulations of quantum-supremacy circuits do not require specialized systems, and (ii) monetary cost can be estimated for cloud-based simulations. Within a given physical-time span, supercomputers might be able to simulate larger quantum circuits than those we simulated, but perhaps at a much greater cost.

The Schrödinger-Feynman simulation that we developed compares favorably to competitors:

- Stabilizer-based methods [28, ?] are exponentially sensitive to the number of non-Clifford gates, whereas diagonal gates are easy for our approach. To this end, simulations in [?] involve circuits with up to 60 such gates, whereas the circuits we simulate include hundreds of them.

---

[14]A similar decomposition can be written out with the projections appearing in the second tensor components.

- Bucket-elimination methods [13, 16] handle diagonal gates well, but so far produce one amplitude at a time, whereas we can produce millions. Additionally, bucket-elimination methods have not yet been evaluated on revised quantum-supremacy benchmarks from Section 5, which should be harder for this approach.

Stabilizer-based techniques have the advantage of better handling non-nearest-neighbor circuits, whereas this generalization would impact bucket elimination methods more than it would impact our approach. However, leading quantum computers are consistent with the nearest-neighbor architecture.

Our work emphasizes rapid progress in simulation algorithms and its implications for quantum supremacy experiments, as follows.

- Limited-depth quantum circuits now admit surprisingly efficient simulation on cloud computers.

- Deep quantum circuits without error correction appear out of reach for both simulation (due to exponentially-growing resource requirements) and quantum chips (due to the exponential accumulation of gate errors). Quantum error correction is currently out of reach for near-term quantum computers [3].

- Schrödinger-Feynman hybrid simulation used in our work favors conventional computers over GPUs and does not require fast interconnect found in supercomputers. It can be cast into an Internet-scale `SETI@Home`-style effort, where transient participants all over the world contribute CPU time on their computers to a joint effort.

- Approximate simulation of quantum circuits is surprisingly efective in practice [15], and this raises the bar for quantum supremacy experiments because low quantum circuit fidelity (due to gate errors) facilitates efficiency gains in approximate simulation. To this end, quantum computers incur stochastic errors in qubit initialization, measurement, one- and two-qubit gates, and gate errors tend to increase with the number of qubits. In contrast, our simulation algorithms incur deterministic inaccuracies in only a very small fraction of two-qubit gates, and the complexity of simulation scales linearly with circuit fidelity f.

- We show how to interactively validate the results of quantum-supremacy simulations with lower-fidelity simulations.

- Second-generation benchmarks, with iSWAP gates instead of CZ gates, appear significantly harder to simulate, yet implementable on a quantum computer. We estimate that changing CZ gates to iSWAP gates reduces circuit depth attainable by leading simulation methods approximately two-fold. In this context, two-qubit gate fidelity remains a key parameter to demonstrate during many-qubit operation.

Pricing quantum-supremacy simulations on cloud-computing services in terms of *cost per amplitude* should facilitate fair comparisons between different hardware types and can reflect the lower cost of idling general-purpose resources. Comparisons with quantum computers by cost are also possible and can help (*i*) making a long-term case for commercial quantum computers, (*ii*) pricing quantum computing resources. Such comparisons must account not only for the resources used by each simulation and each quantum computer run, but also the number of runs required to produce trustworthy results.[16] When sampling outputs on a quantum computer based on superconducting qubits, reported times are around 45 *ns* per cycle with CZ gates, 25 *ns* per cycle with only single-qubit gates [25, 27]. Readout can be as fast as 140 *ns* [35], and qubits can be initialized to a known state in a few hundred *ns* [36, 37]. Thus, comparisons to simulation can be performed once we know circuit fidelity values for 49 or 56 qubits, as well as the cost of quantum computing resources. As both quantum and classical computing technologies continue to improve and become less costly, quantum supremacy remains a moving target and will require careful comparisons.

Acknowledgements

---

[15] Complexity-theoretic arguments against the existence of classical polynomial-time approximation algorithms for this sampling problem [5] do not conflict with the algorithm in our work, as our algorithm requires exponential time.

[16] After saving one million probabilities, one can produce a number of bitstring samples, whereas a quantum computer produces one sample per run.

References

[1] Aaronson, S. & Arkhipov, A. The computational complexity of linear optics. In *STOC*, 333–342 (ACM, 2011).

[2] Bremner, M. J., Montanaro, A. & Shepherd, D. J. Average-case complexity versus approximate simulation of commuting quantum computations. *Phys. Rev. Lett.* **117**, 080501 (2016).

[3] Preskill, J. Quantum computing and the entanglement frontier (2012). 25th Solvay Conf.

[4] Bremner, M. J., Montanaro, A. & Shepherd, D. J. Achieving quantum supremacy with sparse and noisy commuting quantum computations. *Quantum* **1**, 8 (2017).

[5] Boixo, S. *et al.* Characterizing quantum supremacy in near-term devices. *Nat. Phys.* **14**, 595 (2018). arXiv:1608.00263.

[6] Harrow, A. W. & Montanaro, A. Quantum computational supremacy. *Nature* **549**, 203 (2017).

[7] Aaronson, S. & Chen, L. Complexity-theoretic foundations of quantum supremacy experiments. *arXiv:1612.05903* (2016).

[8] Neill, C. *et al.* A blueprint for demonstrating quantum supremacy with superconducting qubits. *arXiv:1709.06678* (2017).

[9] Bouland, A., Fefferman, B., Nirkhe, C. & Vazirani, U. Quantum Supremacy and the Complexity of Random Circuit Sampling. *arXiv:1803.04402* (2018). ArXiv:1803.04402.

[10] De Raedt, K. *et al.* Massively parallel quantum computer simulator. *Computer Physics Communications* **176**, 121–136 (2007).

[11] Häner, T. & Steiger, D. S. 0.5 Petabyte Simulation of a 45-Qubit Quantum Circuit. *arXiv:1704.01127* (2017).

[12] Pednault, E. *et al.* Breaking the 49-qubit barrier in the simulation of quantum circuits. *arXiv:1710.05867* (2017).

[13] Boixo, S., Isakov, S. V., Smelyanskiy, V. N. & Neven, H. Simulation of low-depth quantum circuits as complex undirected graphical models. *arXiv:1712.05384* (2017).

[14] Chen, Z.-Y. *et al.* 64-qubit quantum circuit simulation. *arXiv:1802.06952* (2018).

[15] Li, R., Wu, B., Ying, M., Sun, X. & Yang, G. Quantum supremacy circuit simulation on Sunway TaihuLight. *arXiv:804.04797* (2018).

[16] Chen, J., Zhang, F., Huang, C., Newman, M. & Shi, Y. Classical Simulation of Intermediate-Size Quantum Circuits. *arXiv:1805.01450* (2018).

[17] De Raedt, H. *et al.* Massively parallel quantum computer simulator, eleven years later. *arXiv:1805.04708* (2018).

[18] IBM Announces Advances to IBM Q Systems & Ecosystem, IBM Press Release (2017).

[19] Quantum Computing. URL `https://newsroom.intel.com/press-kits/quantum-computing/`.

[20] A Preview of Bristlecone, Googles New Quantum Processor, Google AI Blog (2018).

[21] Neville, A. *et al.* Classical boson sampling algorithms with superior performance to near-term experiments. *Nat. Phys.* **13**, 1153–1157 (2017).

[22] Bernstein, E. & Vazirani, U. Quantum complexity theory. *SIAM Journal on Computing* **26**, 1411–1473 (1997).

[23] Markov, I. L. & Shi, Y. Simulating quantum computation by contracting tensor networks. *SICOMP* **38**, 963–981 (2008).

[24] Boixo, S. & Neill, C. The question of quantum supremacy. *Google AI Blog* (2018). Available on GitHub at `https://github.com/sboixo/GRCS`.

[25] Barends, R. *et al.* Superconducting quantum circuits at the surface code threshold for fault tolerance. *Nature* **508**, 500–503 (2014).

[26] Barends, R. *et al.* Digitized adiabatic quantum computing with a superconducting circuit. *Nature* **534**, 222–226 (2016).

[27] Kelly, J. *et al.* State preservation by repetitive error detection in a superconducting quantum circuit. *Nature* **519**, 66–69 (2015).

[28] Bravyi, S. & Gosset, D. Improved classical simulation of quantum circuits dominated by Clifford gates. *Phys. Rev. Lett.* **116**, 250501 (2016).

[29] Bravyi, S. *et al.* Simulation of quantum circuits by low-rank stabilizer decompositions. arXiv:1808.00128 (2018).

[30] Viamontes, G. F., Markov, I. L. & Hayes, J. P. *Quantum circuit simulation* (Springer Science & Business Media, 2009).

[31] Høyer, P., Neerbek, J. & Shi, Y. Quantum complexities of ordered searching, sorting, and element distinctness. *Algorithmica* **34**, 429–448 (2002).

[32] Gottesman, D. The heisenberg representation of quantum computers. *arXiv: quant-ph/9807006* (1998).

[33] Nielsen, M. A. *et al.* Quantum dynamics as a physical resource. *Phys. Rev. A* **67**, 052301 (2003).

[34] Neeley, M. *et al.* Generation of three-qubit entangled states using superconducting phase qubits. *Nature* **467**, 570 (2010).

[35] McKay, D. C. *et al.* Universal gate for fixed-frequency qubits via a tunable bus. *Phys. Rev. Applied* **6**, 064007 (2016).

[36] Jeffrey, E. *et al.* Fast accurate state measurement with superconducting qubits. *Phys. Rev. Lett.* **112**, 190504 (2014).

[37] Reed, M. *et al.* Fast reset and suppressing spontaneous emission of a superconducting qubit. *Appl. Phys. Lett.* **96**, 203110 (2010).

[38] Magnard, P. *et al.* Fast and unconditional all-microwave reset of a superconducting qubit. *arXiv:1801.07689* (2018).

Appendix A: Comparisons to Microsoft QDK

To put the capabilities of our simulator in perspective, we compared it to the Microsoft Quantum Development Kit (QDK) v0.2.1806.3001 (June 30 2018).[17] This software includes a back-end circuit simulator and front-end support for the Q# language, integrated with Microsoft Visual Studio and available on Windows, MacOS and Linux. Originally released to the public in the end of 2017, the package underwent significant improvements in February and June 2018, particularly in simulation speed. According to Microsoft, the QDK has been used by tens of thousands of developers worldwide.

To compare the QDK simulator to our simulator Rollright, we used revised (v2) circuit benchmarks described in Section 5 and converted circuit descriptions into the Q# language. In particular, circuit depth $1+26+1$ indicates layers of Hadamard gates at the beginning and at the end. Comparisons were performed on a MacBook Pro 2017 with 16 GiB RAM and Intel Core i7-7700HQ CPU (2.80GHz). Results are reported in Table 4. To exclude code segments from memory comparisons, we first measured max resident memory for each simulator on the 16-qubit benchmark and then used those measurements as baselines. For 24- and 25-qubit benchmarks we report the difference between max resident memory and the respective baseline. Given that the baseline includes memory used by 16-qubit data structures, these estimates are a little lower than the actual data structures, but the inaccuracy is close to 1 MiB and removed by rounding, as can be seen from Rollright data — 128 MiB and 256 MiB are the actual sizes of amplitude vectors in Rollright. We note that Q# uses double-precision floating-point numbers to represent amplitudes, whereas Rollright uses single-precision numbers (while keeping track of accuracy through higher-accuracy norm computations). Hence, the memory-usage ratio should be at least 2.0 on any platform. The runtime ratios are more remarkable and are explained by two factors: (*i*) a certain slowdown due to the use of the Microsoft .NET framework, (*ii*) algorithmic differences. In particular, algorithmic differences explain why the runtime ratio grows with the number of qubits.

We note that for circuits with 30 qubits and fewer, the best runtimes *on a laptop* are attained with the basic Schrödinger capability of our simulator. However, the Microsoft QDK simulator required more than the 16 GiB memory available (Rollright used a little over 8 GiB). We therefore installed Microsoft QDK on an 18-core Linux server with sufficient memory and observed that it used all 72 available threads. We first configured Rollright in the Schrödinger mode. As shown in Table 4, Rollright is 19× faster than the QDK simulator with a 3× memory advantage. The ratios remain similar for circuits of larger depth, and the memory ratio does not depend on how many threads Rollright uses. We then evaluated Rollright in the Schrödinger-Feynman mode using the even (30=15+15) cut, with 18 parallel processes and four threads per process. Rollright was 31× faster and used 886× less memory. The memory ratio depends linearly on

---

[17]Downloaded from `https://www.microsoft.com/en-us/quantum/development-kit` in July 2018.

Circuits		Gates		Microsoft QDK		U. of Michigan Rollright			QDK/Rollright	
(depth $1+26+1$)		All	2-q	Time (s)	Memory (MiB)	Mode	Time	Memory	Time	Memory
MacBook Pro 2017 — MacOS High Sierra: 16 GiB, Intel Core i7-7700HQ (2.80GHz) 4 cores 8 threads										
`inst_4x4_27_5`		274	78	2.44	—	S	< 0.1	—	—	—
`inst_4x6_27_0`		417	123	14.22	452	S	1.19	128	11.95	3.53
`inst_5x5_27_5`		435	130	27.47	837	S	1.68	256	16.35	3.27
`inst_5x6_27_5`		524	161	—	out of memory	S	72	8192	—	—
Server — Ubuntu Linux: 144 GiB, Intel Xeon Platinum 8124M (3.00GHz) 18 cores, 72 threads										
`inst_5x6_27_5`		524	161	400.72	23930.58	**S**	24.4	8192	**16.42**	**2.92**
`inst_5x6_27_5`		524	161	400.72	23930.58	**S-F**	12.95	27.00	**30.94**	**886.32**

Table 4:   Comparisons of our simulator Rollright in the Schrödinger mode to the simulator from Microsoft QDK v0.2.1806.3001 (June 30 2018) on 24-, 25- and 30-qubit benchmarks from Google (v2) from Section 5, performed on a laptop and a mid-range server. Memory usage is reported as the increase in max resident memory relative to the baseline 16-qubit simulation, to exclude code-segment size. For 30-qubit simulation, Rollright demonstrates greater improvements (over Microsoft QDK) in the Schrödinger-Feynman mode with 18 parallel processes than in the single-process Schrödinger mode.

how many simultaneous processes Rollright uses. The runtime ratio grows exponentially with circuit depth (specifically, with the number of xCZ gates). Detailed simulation logs are shown in appendices.

While Rollright significantly outperforms the Microsoft SDK simulator in our comparisons, we consider only quantum-supremacy benchmarks, whereas Microsoft developers may have had different priorities and continue improving their software. Our algorithmic contributions can be useful when simulating many families of quantum circuits, but the technical effort necessary to demonstrate such applications is beyond the scope of this paper. Moreover, Microsoft SDK offers value in areas unrelated to circuit simulation.

Appendix B1: 30-qubit depth $1+26+1$ simulation on a laptop

This Schrödinger-mode simulation uses the circuit `inst_5x6_27_5` publicly available from [24].

```
(C) 2017, 2018  Regents of the University of Michigan
Rollright ver 2.3 - a quantum circuit simulator
Igor L. Markov and Aneeqa Fatima

Hostname : Aneeqa-Macbook Darwin Kernel Version 17.4.0: Sun Dec 17 09:19:54 PST 2017
CPU model name : Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
CPU cores : 4
Hardware threads : 8
Max threads per process : 8
L2 cache size : 262144
L3 cache size : 6291456
CPU instructions width : popcnt:4, sse4.2:256, avx:512, avx2:1024
Using instructions : AVX-2, popcnt

Compiler : gcc 7.3.0
Compiled on : Jul 24 2018 02:04:06
Executed on : 7/27/2018 11:0:37
Size of complex : 8 B

Verbosity : 3
Circuit file : inst_5x6_27_5.txt
Circuit type : Google
Qubits : 30  Gates : 524 (161 two q gates) Cycles : 28
Simulation type : full state-vector
Low-value qubits : 15 q
Layers simulated : H (2), CZ & T (13), X & Y & H (13)
```

```
State representation size : 8 GiB
Norm : 1
Mean entropy : 22 Cross entropy : 23.2
Probabilities : 2.18e+23(min), 8.64e+33(max), 9.31e-10(avg)
Log_2 (max / min) = 35.2
Avg inaccuracy per probability > 5.89e-17 (6.32e-06%)

amp[3]   = -1.96742e-05 + 1.59778e-05j
amp[1/4] = 3.55476e-08 + 4.40849e-06j
amp[1/2] = -1.06781e-05 + 1.58316e-06j
amp[3/4] = 1.256e-06 + 4.03744e-05j
amp[-3]  = 3.07262e-05 - 3.1141e-05j

Runtime (71 s total) by category
     Initial H (30)                               : 4.06 s   =  5.72%
     Last H (12)                                  : 5.16 s   =  7.27%
     CZ & T (280), Low X & Y (90) & H (14)        : 21.4 s   =  30.1%
     High X & Y (94) & H (4)                       : 39.5 s   =  55.7%
     Rescaling passes (1)                         : 0.881 s  =  1.24%
     Storing amps                                 : 0.0023 s =  0.00324%
                                                               -------
     Total                                                    94.3%

Average time per gate : 0.135 s
```

Appendix B2: 30-qubit depth $1+26+1$ simulation on a server

This Schrödinger-mode simulation uses the circuit `inst_5x6_27_5` publicly available from [24].

```
(C) 2017, 2018  Regents of the University of Michigan
Rollright ver 2.3 - a quantum circuit simulator
Igor L. Markov and Aneeqa Fatima

Hostname : Linux ip-172-31-24-65 4.4.0-1049 #58-Ubuntu SMP Fri Jan 12 23:17:09 UTC 2018
CPU model name : Intel(R) Xeon(R) Platinum 8124M CPU @ 3.00GHz
cpu cores : 18
Hardware threads : 72
Max threads per process : 72
MemTotal:       144156168 kB
L3 cache size : 25344 KB
CPU instructions width : popcnt:4, sse4.2:256, avx:512, avx2:1024
Using instructions : AVX-2, popcnt

Compiler : gcc 7.2.0
Compiled on : Jul 27 2018 06:58:56
Executed on : 7/27/2018 14:59:50
Size of complex : 8 B

Verbosity : 3
Circuit file : inst_5x6_27_5.txt
Circuit type : Google
Qubits : 30  Gates : 524 (161 two q gates) Cycles : 28
Simulation type : full state-vector
Low-value qubits : 15 q
Layers simulated : H (2), CZ & T (13), X & Y & H (13)

State representation size : 8 GiB
Norm : 1
Mean entropy : 22 Cross entropy : 23.2
Probabilities : 2.18e+23(min), 8.64e+33(max), 9.31e-10(avg)
```

```
Log_2 (max / min) = 35.2
Avg inaccuracy per probability > 5.89e-17 (6.32e-06%)


amp[3]   = -1.96742e-05 + 1.59778e-05j
amp[1/4] = 3.55476e-08 + 4.40849e-06j
amp[1/2] = -1.06781e-05 + 1.58316e-06j
amp[3/4] = 1.256e-06 + 4.03744e-05j
amp[-3]  = 3.07262e-05 - 3.1141e-05j


Runtime (24.4 s total) by category
Initial H (30)                          : 0.432 s    =  1.77%
Last H (12)                             : 1.85 s     =  7.6%
CZ & T (280), Low X & Y (90) & H (14)   : 3.51 s     =  14.4%
High X & Y (94) & H (4)                 : 18.4 s     =  75.3%
Rescaling passes (1)                    : 0.226 s    =  0.925%
Storing amps                            : 3.65e-05 s =  0.00015%
                                                       -------
Total                                                  98.2%


Average time per gate : 0.0465 s
```

Appendix B3: 30-qubit depth $1 + 26 + 1$ cloud simulation

This Schrödinger-Feynman simulation uses the circuit `inst_5x6_27_5` publicly available from [24].

```
(C) 2017, 2018  Regents of the University of Michigan
Rollright ver 2.3 - a quantum circuit simulator
Igor L. Markov and Aneeqa Fatima


Hostname : Linux ip-172-31-24-65 4.4.0-1049 #58-Ubuntu SMP Fri Jan 12 23:17:09
CPU model name : Intel(R) Xeon(R) Platinum 8124M CPU @ 3.00GHz
cpu cores : 18
Hardware threads : 72
Max threads per process : 4
MemTotal:       144156168 kB
L3 cache size : 25344 KB
CPU instructions width : popcnt:4, sse4.2:256, avx:512, avx2:1024
Using instructions : AVX-2, popcnt


Compiler : gcc 7.2.0
Compiled on : Jul 27 2018 06:58:56
Executed on : 7/27/2018 15:26:49
Size of complex : 8 B
Verbosity : 3


Circuit file : inst_5x6_27_5.txt
Circuit type : Google
Qubits : 30  Gates : 524 (161 two q gates) Cycles : 28
Qubit grid 5x6 with 2 blocks
     0^   1^   2^   3_   4_   5_
     6^   7^   8^   9_  10_  11_
    12^  13^  14^  15_  16_  17_
    18^  19^  20^  21_  22_  23_
    24^  25^  26^  27_  28_  29_
 Cross-gates : 5


Simulation type : sum of tensor products / single cut
Cut : vertical 15q + 15q (17 xCZ gates)
Simulating xCZ gates : using projection-based branches
xCZ path breakdown : 10p + 5r + 2b (cycle breakdown : 19p + 8r + 1b)
```

```
Low-value qubits : 8 q, 8 q
Requested num amps : 1000

Multi-process simulation :
    1024 processes (4 threads each) over 1 node in 18 batches per node
    State representation size : 512 KiB
    Peak memory : 27648.0 KiB
    Layers simulated : H (193), CZ & T (330), X & Y & H (330)
    Layers breakdown : 11p + 192r + 128b
     Batch stats :
          Avg user time : 44.429 s
          Wallclock : 12.725 s (avg), 12.91 s (max)
          Avg CPU utilization : 348.36% (87.09% per thread)
          Avg resident size : 5.816 MiB
          Avg page faults : 0.018 (major), 5795.459 (minor)
      Billable runtime : 3.586e-03 hrs (3.586e-06 hrs per amp)

amp[3]   = -1.967417e-05+1.597779e-05j
amp[1/4] = 3.556267e-08+4.408506e-06j
amp[1/2] = -1.067803e-05+1.583226e-06j
amp[3/4] = 1.256007e-06+4.037441e-05j
amp[-3]  = 3.072615e-05-3.114105e-05j

Avg runtime (0.2 s total) per process by category
    Initial H (30)                     : 0.00328 s   = 1.636%
    Last H (14)                        : 0.007 s     = 3.391%
    xCZ (17)                           : 0.009 s     = 4.505%
    CZ & T (263), Low X & Y (84) & H (12) : 0.038 s   = 19.066%
    Single X (9) & Y (11)              : 0.008 s     = 4.021%
    High X & Y (80) & H (4)            : 0.007 s     = 3.51%
    Rescaling passes (32)              : 0.001 s     = 0.355%
    Copying (127)                      : 0.006 s     = 3.071%
    Storing amps                       : 0.047 s     = 23.453%
                                                       ----------
    Total                                              63.007%

Avg time per gate : 3e-06 s
Per-process runtime breakdown :
     Prefix         : 0.068 s = 34.159%
     Branches       : 0.137 s = 68.47%
     Memory map I/O : 2e-05 s = 0.00799%
```

Appendix C: "Easy" 42-qubit depth $1 + 25 + 1$ simulation

This Schrödinger-Feynman simulation uses the circuit `inst_7x6_26_0` publicly available from [24], Such revised benchmarks are harder to simulate than older quantum-supremacy benchmarks (see Section 5).

```
(C) 2017, 2018  Regents of the University of Michigan
Rollright ver 2.3 - a quantum circuit simulator
Igor L. Markov and Aneeqa Fatima

Hostname : Linux boixo-rr-h96-5 4.13.0-45-generic #50-Ubuntu
CPU model name : Intel(R) Xeon(R) CPU @ 2.00GHz
cpu cores : 24
Hardware threads : 96
Max threads per process : 4
MemTotal:       89073048 kB
L3 cache size : 56320 KB
Hugepagesize:       2048 kB
```

```
HugePages_Total:        0
HugePages_Free:         0
CPU instructions width : popcnt:4, sse4.2:256, avx:512, avx2:1024
Using instructions : AVX-2, popcnt


Compiler : gcc 7.2.0
Compiled on : Jul 24 2018 02:36:06
Executed on : 7/29/2018 9:16:36
Size of complex : 8 B


Verbosity : 3
Circuit file : inst_7x6_26_0
Circuit type : Google
Qubits : 42  Gates : 711 (224 two q gates) Cycles : 27
Qubit grid 7x6 with 2 blocks
      0^    1^    2^    3^    4^    5^
      6^    7^    8^    9^   10^   11^
     12^   13^   14^   15^   16^   17^
     18^   19^   20^   21^   22^   23^
     24_   25_   26_   27_   28_   29_
     30_   31_   32_   33_   34_   35_
     36_   37_   38_   39_   40_   41_
 Cross-gates : 6


Simulation type : sum of tensor products / single cut
Cut : horizontal 24q + 18q (18 xCZ gates)
Simulating xCZ gates : using projection-based branches
xCZ path breakdown : 6p + 6r + 6b (cycle breakdown : 13p + 8r + 6b)
Low-value qubits : 12 q, 9 q
Requested num amps : 1000000

Multi-process simulation :
     64 processes (4 threads each) over 1 node in 23 batches per node
     State representation size : 130 MiB
     Peak memory : 8970.0 MiB
     Layers simulated : H (4097), CZ & T (16775), X & Y & H (12679)
     Layers breakdown : 8p + 384r + 16384b
     Batch stats :
         Avg user time : 10.26 hrs
         Wallclock : 5.534 hrs (avg), 6.016 hrs (max)
         Avg CPU utilization : 185.766% (46.441% per thread)
         Avg resident size : 435.466 MiB
         Avg page faults : 0.406 (major), 14708868.438 (minor)
     Billable runtime : 6.016e+00 hrs (6.016e-06 hrs per amp)
     Avg zero count
         1st Checkpoint : A = 0 (0.0%), B = 0 (0.0%)
         2nd Checkpoint : A = 0 (0.0%), B = 0 (0.0%)

amp[3]    = 2.998598e-07-2.862689e-07j
amp[1/4]  = 2.248877e-07+1.877150e-07j
amp[1/2]  = 2.425415e-07-1.685559e-08j
amp[3/4]  = -4.233980e-07-7.303483e-08j
amp[-3]   = 4.674360e-07-2.024549e-07j

Avg runtime (7147.031 s total) per process by category
     Initial H (42)                          : 0.03196 s    = 0.0%
     Last H (22)                             : 411.156 s    = 5.753%
     xCZ (18)                                : 168.734 s    = 2.361%
     CZ & T (363), Low X & Y (108) & H (20)  : 917.344 s    = 12.835%
     Single X (7) & Y (5)                    : 95.763 s     = 1.34%
```

```
    High X & Y (126)                   : 635.188 s    = 8.887%
    Rescaling passes (64)              : 1.199 s      = 0.017%
    Copying (4095)                     : 131.828 s    = 1.845%
    Storing amps                       : 4620.312 s   = 64.647%
                                                        ----------
    Total                                             97.685%


Avg time per gate : 0.002454 s
Simulation per process runtime breakdown :
    Prefix            : 175.703 s = 2.458%
    Branches          : 7100.469 s = 99.349%
    Memory map I/O     : 0.01171 s = 0.00016%
```

Appendix D: 64-qubit depth $1+22$ simulation

This simulation uses a circuit with 8×8 qubits and depth $1+22$. For comparison with results in [14], we use the ordering of CZ gates as that work, which is approximately 16 times easier for this simulation method at this depth.

```
(C) 2017, 2018  Regents of the University of Michigan
Rollright ver 2.3 - a quantum circuit simulator
Igor L. Markov and Aneeqa Fatima

Hostname : Linux boixo-rr-hmm32-36 4.13.0-45-generic #50-Ubuntu
CPU model name : Intel(R) Xeon(R) CPU @ 2.00GHz
CPU cores : 16
Hardware threads : 32
Max threads per process : 32
MemTotal:       214601516 kB
L3 cache size : 56320 KB
CPU instructions width : popcnt:4, sse4.2:256, avx:512, avx2:1024
Using instructions : AVX-2, popcnt

Compiler : gcc 7.2.0
Compiled on : Jun 29 2018 21:51:16
Executed on : 6/30/2018 18:20:54
Size of complex : 8 B


Verbosity : 3
Circuit file : inst_8_8_22_0_chen
Circuit type : Google
Qubits : 64  Gates : 848  Cycles : 23
Qubit grid 8x8 with 2 blocks
    0^   1^   2^   3^   4^   5^   6^   7^
    8^   9^   10^  11^  12^  13^  14^  15^
    16^  17^  18^  19^  20^  21^  22^  23^
    24^  25^  26^  27^  28^  29^  30^  31^
    32_  33_  34_  35_  36_  37_  38_  39_
    40_  41_  42_  43_  44_  45_  46_  47_
    48_  49_  50_  51_  52_  53_  54_  55_
    56_  57_  58_  59_  60_  61_  62_  63_
 Cross-gates : 8


Simulation type : sum of tensor products / single cut
Cut : horizontal 32q + 32q (16 xCZ gates)
Simulating xCZ gates : using projection-based branches
xCZ path breakdown : 8p + 4r + 4b (cycle breakdown : 16p + 0r + 7b)
Low-value qubits : 16 q, 16 q
```

```
Requested num amps : 16384

Multi-process simulation :
      256 processes (32 threads each) over 64 nodes in 1 batch per node
      State representation size : 64 GiB
      Peak memory : 12288.0 GiB (192.0 GiB per node)
      Layers simulated : H (1), CZ & T (1048), X & Y (1048)
      Layers breakdown : 9p + 16r + 1024b
      Batch stats :
           Avg user time : 254.053 hrs
           Wallclock : 14.726 hrs (avg), 18.593 hrs (max)
           Avg CPU utilization : 1734.633% (54.207% per thread)
           Avg resident size : 187.505 GiB
           Avg page faults : 0.0 (major), 301993529.113 (minor)
      Billable runtime : 1.190e+03 hrs (7.263e-02 hrs per amp)

amp[3]   = -3.572132e-11-1.877656e-10j
amp[1/4] = 2.245370e-10+2.368543e-10j
amp[1/2] = 2.019670e-10+1.266517e-10j
amp[3/4] = 7.522537e-11+8.860908e-11j
amp[-3]  = -1.860613e-10-2.177499e-10j

Avg runtime (13207.812 s total) per process by category
      Initial H (64)                 : 1.8916 s      = 0.014%
      xCZ (16)                       : 454.203 s     = 3.439%
      CZ & T (480), Low X & Y (138)  : 3210.234 s    = 24.306%
      Single X (7) & Y (3)           : 2404.062 s    = 18.202%
      High X & Y (140)               : 6346.875 s    = 48.054%
      Rescaling passes (1)           : 1.872 s       = 0.014%
      Copying (255)                  : 699.098 s     = 5.293%
      Storing amps                   : 6.254 s       = 0.047%
                                                      ----------
      Total                                          99.369%

Avg time per gate : 0.060839 s
Simulation per process runtime breakdown :
Prefix                 : 857.234 s    = 6.49%
Branches               : 13005.469 s  = 98.468%
Memory mapped I/O      : 0.00015 s    = 0.0%
```

Appendix E1: 49-qubit depth $1 + 39 + 1$ simulation with fidelity 1%

This simulation uses the circuit `inst_7x7_40_0` publicly available from [24].

```
(C) 2017, 2018  Regents of the University of Michigan
Rollright ver 2.2 - a quantum circuit simulator
Igor L. Markov and Aneeqa Fatima

Hostname : Linux boixo-rr-h32-333 4.13.0-43-generic #48-Ubuntu
CPU model name : Intel(R) Xeon(R) CPU @ 2.30GHz
CPU cores : 16
Hardware threads : 32
Max threads per process : 8
MemTotal:        123768660 kB
L3 cache size : 46080 KB
CPU instructions width : popcnt:4, sse4.2:256, avx:512, avx2:1024
```

```
Using instructions : AVX-2, popcnt

Compiler : gcc 7.2.0
Compiled on : May 28 2018 21:20:26
Executed on : 5/29/2018 9:37:17
Size of complex : 8 B


Verbosity : 3
Circuit file : inst_7x7_40_0
Circuit type : Google
Qubits : 49  Gates : 1252  Cycles : 41
Qubit grid 7x7 with 2 blocks
     0^   1^   2^   3^   4^   5^   6^
     7^   8^   9^  10^  11^  12^  13^
    14^  15^  16^  17^  18^  19^  20^
    21^  22^  23^  24^  25^  26^  27^
    28_  29_  30_  31_  32_  33_  34_
    35_  36_  37_  38_  39_  40_  41_
    42_  43_  44_  45_  46_  47_  48_
 Cross-gates : 7


Simulation type : sum of tensor products / single cut
Cut : horizontal 28q + 21q (31 xCZ gates)
Requested end-to-end circuit fidelity: 0.01
Approximation type : pruned xCZ branches
Simulating xCZ gates : using projection-based branches
xCZ path breakdown : 24p + 4r + 3b (cycle breakdown : 33p + 4r + 4b)
Low-value qubits : 14 q, 11 q
Requested num amps : 1000000

Multi-process simulation :
     167773 processes (8 threads each) in 4 batch(es) over 617 node(s)
     State representation size : 2.02 GiB
     Peak memory : 14956.08 GiB (24.24 GiB per node)
     Layers simulated : H (129), CZ & T (465), X & Y & H (337)
     Layers breakdown : 18p + 64r + 384b
     Batch stats :
         Avg user time : 249090.762 s
         Wallclock : 64480.76 s (avg), 126850.07 s (max)
         Avg CPU utilization : 390.411% (48.801% per thread)
         Avg resident size : 5.976 GiB
         Avg page faults : 0.057 (major), 9480745.407 (minor)
     Billable runtime : 2.174e+04 hrs (2.174e-02 hrs per amp)

amp[3]   = -2.017607e-09-1.396589e-09j
amp[1/4] = 1.534751e-10-2.899289e-09j
amp[1/2] = -4.921908e-09+2.984601e-09j
amp[3/4] = 4.053322e-10-2.020723e-09j
amp[-3]  = 1.646911e-10-2.048092e-09j

Avg runtime (945.34 s total) per process by category
     Initial H (49)                           : 0.18201 s   = 0.019%
     Last H (25)                              : 233.173 s   = 24.665%
     xCZ (31)                                 : 29.88 s     = 3.161%
     CZ & T (665), Low X & Y (220) & H (24) : 206.859 s   = 21.882%
     Single X (3) & Y (7)                     : 39.322 s    = 4.16%
     High X & Y (228)                         : 197.519 s   = 20.894%
     Rescaling passes (9)                     : 0.367 s     = 0.039%
     Copying (144)                            : 41.846 s    = 4.427%
     Storing amps                             : 184.067 s   = 19.471%
```

```
                                                ----------
     Total                                      98.717%

Avg time per gate : 0.005899 s
Simulation per process runtime breakdown :
     Prefix              : 115.664 s = 12.235%
     Branches            : 826.921 s = 87.473%
     Memory mapped I/O   : 0.00968 s = 0.00102%
```

Appendix E2: 49-qubit depth $1 + 40 + 1$ simulation with fidelity 0.51%

This simulation uses the circuit `inst_7x7_41_0` publicly available from [24].

```
(C) 2017, 2018  Regents of the University of Michigan
Rollright ver 2.2 - a quantum circuit simulator
Igor L. Markov and Aneeqa Fatima

Hostname : Linux boixo-rr-hc32-28 4.13.0-43-generic #48-Ubuntu
CPU model name : Intel(R) Xeon(R) CPU @ 2.00GHz
CPU cores : 16
Hardware threads : 32
Max threads per process : 8
MemTotal:       29631356 kB
L3 cache size : 56320 KB
CPU instructions width : popcnt:4, sse4.2:256, avx:512, avx2:1024
Using instructions : AVX-2, popcnt

Compiler : gcc 7.2.0
Compiled on : May 28 2018 21:20:26
Executed on : 6/2/2018 21:21:34
Size of complex : 8 B

Verbosity : 3
Circuit file : inst_7x7_41_0
Circuit type : Google
Qubits : 49  Gates : 1280  Cycles : 42
Qubit grid 7x7 with 2 blocks
      0^    1^    2^    3^    4^    5^    6^
      7^    8^    9^   10^   11^   12^   13^
     14^   15^   16^   17^   18^   19^   20^
     21^   22^   23^   24^   25^   26^   27^
     28_   29_   30_   31_   32_   33_   34_
     35_   36_   37_   38_   39_   40_   41_
     42_   43_   44_   45_   46_   47_   48_
 Cross-gates : 7

Simulation type : sum of tensor products / single cut
Cut : horizontal 28q + 21q (35 xCZ gates)
Requested end-to-end circuit fidelity: 0.0051
Approximation type : pruned xCZ branches
Simulating xCZ gates : using projection-based branches
xCZ path breakdown : 28p + 3r + 4b (cycle breakdown : 37p + 4r + 1b)
Low-value qubits : 14 q, 11 q
Requested num amps : 1000000

Multi-process simulation :
```

```
      1369569 processes (8 threads each) in 4 batch(es) over 625 node(s)
      State representation size : 2.02 GiB
      Peak memory : 15150.0 GiB (24.24 GiB per node)
      Layers simulated : H (145), CZ & T (179), X & Y & H (43)
      Layers breakdown : 20p + 32r + 128b
      Batch stats :
            Avg user time : 615850.294 s
            Wallclock : 132293.755 s (avg), 209608.71 s (max)
            Avg CPU utilization : 541.57% (67.696% per thread)
            Avg resident size : 5.975 GiB
            Avg page faults : 0.008 (major), 5286338.665 (minor)
      Billable runtime : 3.639e+04 hrs (3.639e-02 hrs per amp)

amp[3]   = -2.405457e-09+2.098312e-10j
amp[1/4] = 3.961961e-10+1.482872e-09j
amp[1/2] = 6.435293e-11+1.481697e-09j
amp[3/4] = -7.280204e-10+1.029079e-09j
amp[-3]  = 1.406264e-09-6.950104e-10j

Avg runtime (238.985 s total) per process by category
      Initial H (49)                        : 0.12541 s    = 0.052%
      Last H (25)                           : 90.339 s     = 37.801%
      xCZ (35)                              : 16.716 s     = 6.995%
      CZ & T (678), Low X & Y (224) & H (18) : 17.128 s    = 7.167%
      Single X (6) & Y (5)                  : 3.409 s      = 1.426%
      High X & Y (234) & H (6)              : 20.769 s     = 8.691%
      Rescaling passes (6)                  : 1.121 s      = 0.469%
      Copying (136)                         : 24.359 s     = 10.193%
      Storing amps                          : 56.75 s      = 23.746%
                                                           ----------
      Total                                                96.541%

Avg time per gate : 0.001459 s
Simulation per process runtime breakdown :
      Prefix          : 54.786 s  = 22.924%
      Branches        : 182.966 s = 76.56%
      Memory mapped I/O : 0.00758 s = 0.00317%
```

Appendix E3: 49-qubit depth $1 + 48 + 1$ simulation with fidelity 2^{-22}

This simulation uses the circuit inst_7x7_49_0 publicly available from [24].

```
(C) 2017, 2018  Regents of the University of Michigan
Rollright ver 2.3 - a quantum circuit simulator
Igor L. Markov and Aneeqa Fatima

Hostname : Linux boixo-rr-hc32-28 4.13.0-45-generic #50-Ubuntu
CPU model name  : Intel(R) Xeon(R) CPU @ 2.00GHz
cpu cores       : 16
Hardware threads : 32
Max threads per process : 8
MemTotal:       29631356 kB
L3 cache size   : 56320 KB
CPU instructions width : popcnt:4, sse4.2:256, avx:512, avx2:1024
Using instructions : AVX-2, popcnt
```

```
Compiler : gcc 7.2.0
Compiled on : Jul 24 2018 02:36:06
Executed on : 7/27/2018 7:9:14
Size of complex : 8 B


Verbosity : 3
Circuit file : inst_7x7_49_0
Circuit type : Google
Qubits : 49  Gates : 1516 (504 two q gates) Cycles : 50
Qubit grid 7x7 with 2 blocks
      0^   1^   2^   3^   4^   5^   6^
      7^   8^   9^  10^  11^  12^  13^
     14^  15^  16^  17^  18^  19^  20^
     21^  22^  23^  24^  25^  26^  27^
     28_  29_  30_  31_  32_  33_  34_
     35_  36_  37_  38_  39_  40_  41_
     42_  43_  44_  45_  46_  47_  48_
 Cross-gates : 7


Simulation type : sum of tensor products / single cut
Cut : horizontal 28q + 21q (42 xCZ gates)
Requested end-to-end circuit fidelity: 2.38e-07
Approximation type : pruned xCZ branches
Simulating xCZ gates : using projection-based branches
xCZ path breakdown : 35p + 4r + 3b (cycle breakdown : 45p + 4r + 1b)
Low-value qubits : 14 q, 11 q
Requested num amps : 1000000

Multi-process simulation :
    8192 processes (8 threads each) over 512 nodes in 4 batches per node
    State representation size : 2.02 GiB
    Peak memory : 12410.88 GiB (24.24 GiB per node)
    Layers simulated : H (161), CZ & T (215), X & Y & H (71)
    Layers breakdown : 24p + 64r + 128b
    Batch stats :
        Avg user time : 5475.45 s
        Wallclock : 1074.158 s (avg), 1536.13 s (max)
        Avg CPU utilization : 582.699% (72.837% per thread)
        Avg resident size : 5.985 GiB
        Avg page faults : 0.63 (major), 9500721.534 (minor)
    Billable runtime : 2.185e+02 hrs (2.185e-04 hrs per amp)
    Avg zero count
        1st Checkpoint : A = 0 (0.0%), B = 0 (0.0%)
        2nd Checkpoint : A = 16777216 (6.25%), B = 0 (0.0%)

amp[3]          = -1.921780e-11-2.707243e-12j
amp[1/4]        = 1.118916e-11-1.348977e-11j
amp[1/2]        = -4.073272e-12-9.529898e-13j
amp[3/4]        = -1.650529e-11+9.459437e-12j
amp[-3]         = 2.059435e-11-1.063607e-11j

Avg runtime (265.755 s total) per process by category
      Initial H (49)                          : 0.13483 s    = 0.051%
      Last H (25)                             : 97.202 s     = 36.576%
      xCZ (42)                                : 21.841 s     = 8.219%
      CZ & T (811), Low X & Y (270) & H (18)  : 29.279 s     = 11.017%
      Single X (8) & Y (5)                    : 5.889 s      = 2.216%
      High X & Y (282) & H (6)                : 33.962 s     = 12.779%
      Rescaling passes (2)                    : 0.374 s      = 0.141%
```

```
        Copying (127)                          : 29.357 s    = 11.047%
        Storing amps                           : 35.275 s    = 13.274%
                                                              ----------
        Total                                                 95.319%


Avg time per gate : 0.00137 s
Simulation per process runtime breakdown :
      Prefix              : 119.984 s = 45.148%
      Branches            : 171.701 s = 64.609%
      Memory map I/O       : 0.00753 s = 0.00283%
```

Appendix F: 56-qubit depth $1 + 40 + 1$ simulation with fidelity 0.51%

This simulation uses the circuit `inst_7x8_41_0` publicly available from [24].

```
(C) 2017, 2018  Regents of the University of Michigan
Rollright ver 2.2 - a quantum circuit simulator
Igor L. Markov and Aneeqa Fatima

Hostname : Linux boixo-rr-hs32-28 4.13.0-43-generic #48-Ubuntu
CPU model name : Intel(R) Xeon(R) CPU @ 2.00GHz
CPU cores : 16
Hardware threads : 32
Max threads per process : 8
MemTotal:       123768660 kB
L3 cache size : 56320 KB
CPU instructions width : popcnt:4, sse4.2:256, avx:512, avx2:1024
Using instructions : AVX-2, popcnt

Compiler : gcc 7.2.0
Compiled on : Jun  8 2018 07:28:48
Executed on : 6/20/2018 18:18:13
Size of complex : 8 B

Verbosity : 3
Circuit file : inst_7x8_41_0
Circuit type : Google
Qubits : 56  Gates : 1466  Cycles : 42
Qubit grid 7x8 with 2 blocks
      0^   1^   2^   3^   4_   5_   6_   7_
      8^   9^  10^  11^  12_  13_  14_  15_
     16^  17^  18^  19^  20_  21_  22_  23_
     24^  25^  26^  27^  28_  29_  30_  31_
     32^  33^  34^  35^  36_  37_  38_  39_
     40^  41^  42^  43^  44_  45_  46_  47_
     48^  49^  50^  51^  52_  53_  54_  55_
 Cross-gates : 7

Simulation type : sum of tensor products / single cut
Cut : vertical 28q + 28q (35 xCZ gates)
Requested end-to-end circuit fidelity: 0.0051
Approximation type : pruned xCZ branches
Simulating xCZ gates : using projection-based branches
xCZ path breakdown : 28p + 3r + 4b (cycle breakdown : 37p + 4r + 1b)
Low-value qubits : 14 q, 14 q
Requested num amps : 1000000
```

```
Multi-process simulation :
    1369569 processes (8 threads each) over 625 nodes in 4 batches per node
    State representation size : 4 GiB
    Peak memory : 30000.0 GiB (48.0 GiB per node)
    Layers simulated : H (145), CZ & T (179), X & Y & H (179)
    Layers breakdown : 20p + 32r + 128b
    Batch stats :
        Avg user time : 418.77 hrs
        Wallclock : 87.592 hrs (avg), 140.744 hrs (max)
        Avg CPU utilization : 483.757% (60.47% per thread)
        Avg resident size : 11.774 GiB
        Avg page faults : 0.002 (major), 10508777.493 (minor)
    Billable runtime : 8.796e+04 hrs (8.796e-02 hrs per amp)

amp[3]   = 2.911096e-10-1.236132e-11j
amp[1/4] = 4.302013e-13+1.527466e-10j
amp[1/2] = -7.879150e-11-2.773058e-10j
amp[3/4] = 2.147728e-10-6.191063e-11j
amp[-3]  = -3.276638e-11+5.329280e-11j

Avg runtime (570.988 s total) per process by category
    Initial H (56)                       : 0.22702 s   = 0.04%
    Last H (28)                          : 51.494 s    = 9.018%
    xCZ (35)                             : 41.237 s    = 7.222%
    CZ & T (781), Low X & Y (258) & H (24) : 112.343 s  = 19.675%
    Single X (8) & Y (16)                : 30.526 s    = 5.346%
    High X & Y (256) & H (4)             : 163.514 s   = 28.637%
    Rescaling passes (10)                : 2.547 s     = 0.446%
    Copying (136)                        : 58.704 s    = 10.281%
    Storing amps                         : 99.107 s    = 17.357%
                                           ----------
    Total                                  98.023%

Avg time per gate : 0.003043 s
Simulation per process runtime breakdown :
    Prefix            : 106.885 s = 18.719%
    Branches          : 461.465 s = 80.819%
    Memory mapped I/O : 0.00913 s = 0.0016%
```