Supporting Information: A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware

Benjamin Villalonga,[1,2,3,*] Sergio Boixo,[4,†] Bron Nelson,[2,5,‡] Christopher Henze,[2,§] Eleanor Rieffel,[2,¶] Rupak Biswas,[2,**] and Salvatore Mandrà[2,6,††]

[1] *Institute for Condensed Matter Theory and Department of Physics, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA*
[2] *Quantum Artificial Intelligence Lab. (QuAIL), NASA Ames Research Center, Moffett Field, CA 94035, USA*
[3] *USRA Research Institute for Advanced Computer Science (RIACS), 615 National, Mountain View, California 94043, USA*
[4] *Google Inc., Venice, CA 90291, USA*
[5] *ASRC Federal InuTeq, 7000 Muirkirk Meadows Drive, Suite 100, Beltsville, MD 20705*
[6] *Stinger Ghaffarian Technologies Inc., 7701 Greenbelt Rd., Suite 400, Greenbelt, MD 20770*

Here we present qFlex, a flexible tensor network based simulator of quantum circuits. qFlex can compute both exact amplitudes, a task essential for the verification of the quantum hardware, as well as low fidelity amplitudes, in order to mimic sampling from Noisy Intermediate-Scale Quantum (NISQ) devices. In this work, we focus on random quantum circuits (RQCs) in the range of sizes expected for supremacy experiments. Fidelity f simulations are performed at a cost that is a factor $1/f$ lower than perfect fidelity ones. We benchmark the classical simulation of square lattices and Google's Bristlecone QPU. In addition, we discuss the hardness of the classical simulation of RQCs and give evidence for the higher complexity of the simulation of Bristlecone as compared to other two-dimensional grids with the same number of qubits. Our analysis is supported by extensive simulations on NASA HPC clusters Pleiades and Electra. For the most computationally demanding simulation we had, the two HPC clusters combined reached a peak of 20 PFLOPS (single precision), *i.e.*, 64% of their maximum achievable performance. To date, this numerical computation is the largest in terms of sustained FLOPs and number of nodes utilized ever run on NASA HPC clusters. Finally, we introduce a novel multithreaded, cache-efficient tensor index permutation algorithm of general application.

PACS numbers: 03.65.Aa,03.67.Ac,03.67.Lx,05.45.Pq

A. CONTRACTION OF RQCS ON RECTANGULAR GRIDS

Here we describe the contraction scheme for $I \times J$ grids. To simplify the discussion, we consider $7 \times 7 \times (1 + 40 + 1)$ RQCs. Nonetheless, the same procedure applies to other rectangular circuits. For the $7 \times 7 \times (1+40+1)$ RQCs, we use two cuts ($2^{2\times5}$ paths) in the grid (see Fig. A1), in order to divide it into four tensors, A, B, C, and D, of dimension $2^{6\times5} = 2^{30}$ each. Then, the contraction proceeds as follows.

1) Tensors in region A are contracted onto tensor A, which is path independent; we do the same for tensors in regions pB (partial-B), pC (partial-C), and ppD (partial-partial-D), which are all path independent; for this reason, all A, pB, pC, and ppD are reused over the entire computation. We now iterate over all paths with two nested iterations: the outer one iterates over the right cut, while the inner one iterates over the bottom cut.

2) For each of the 2^5 right paths, tensors in regions B and pD are contracted.

3) Tensors A and B are contracted onto AB; this tensor will be reused over all the inner loop iterations.

4) For each bottom path (given a right path), the tensors on region C and D are contracted.

5) C and D are contracted onto CD.

6) AB and CD are contracted onto a scalar, which is equal to this path's contribution to the amplitude.

From here, the iteration over the inner loop takes us back to step 4), for a different bottom path. After this loop is exhausted, we go back to step 2), for the next right path. After iterating over both loops, we have obtained all path contributions.

Note that there is a large amount of potential reuse of the tensors built at each step. However,

* vlllngc2@illinois.edu
† boixo@google.com
‡ bron.c.nelson@nasa.gov
§ chris.henze@nasa.gov
¶ eleanor.rieffel@nasa.gov
** rupak.biswas@nasa.gov
†† salvatore.mandra@nasa.gov

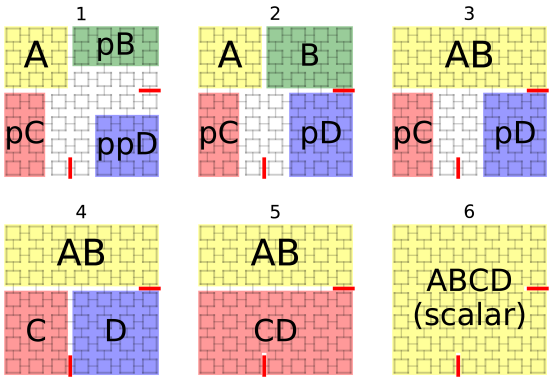

Figure A1. Sketch of the contraction of the $7 \times 7 \times (1 + 40 + 1)$ tensor network with two cuts. The names of the tensors at key steps shown in the contraction are referred to in Section A.

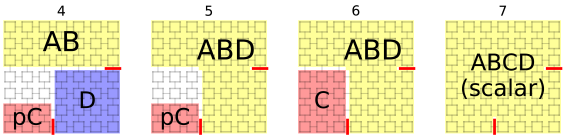

Figure A2. Alternative contraction for the use of fast sampling (see Methods C.2).

taking advantage of already built tensors requires a large amount of memory in order to store all the tensors to be reused. While there is a clear trade-off between tensor reuse and memory usage, in practice we always found the reuse profitable.

Finally, the fast sampling introduced in Methods C.2 can be also applied here, by using a slightly different contraction than the one presented in Fig. A1. More precisely, in Fig. A2 we present the final steps of the alternative contraction. In 4) and 5) the size of the tensor pC is still small, and we focus, for this particular path, in contracting first D with AB onto ABD. This leaves six qubits free (above pC) for the computation of batches of as much as $2^6 = 64$ amplitudes (or more if pC is shrunk further); for each amplitude, contracting the tensors in region C (where pC is reused) and computing the scalar $ABCD$ is left, *i.e.*, steps 6) and 7) of Fig. A2. Once this is done, then we go back to step 5), in order to loop over all bit-strings in the batch. After exhausting this loop, we go back to step 4) to compute the next bottom path. Note that the contraction following this procedure is dominated by the contraction of AB with D; however, the tensor ABD is reused (for a path) for all bit-strings in a batch, and so the computation of a batch of amplitudes adds only a small cost to the computation of a single amplitude.

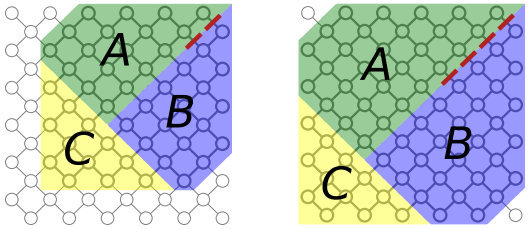

Figure B3. Tensors A, B, and C used in the old contraction schemes used for [Run2] and [Run3] (Bristlecone-48 and -70, respectively).

B. OLD TENSOR CONTRACTIONS FOR BRISTLECONE-48 AND BRISTLECONE-70

The contractions procedures followed for [Run2-3] are presented in Fig. B3. Note that we have identified a faster contraction (about half the runtime) with less memory requirements for Bristlecone-48 and -70, which is described in Methods B.1, and whose runtimes are reported in the main text (see Results) and referred to as [Run2b] and [Run3b], respectively. Note also that the new contraction scheme cannot be adapted to Bristlecone-60, due to its topology. The old contractions are included here for reference, and apply to our released simulation data [1].

C. QUBIT COMPLEXITY OF SQUARE GRIDS AND BRISTLECONE SUB-LATTICES FOR SCHRODINGER-FEYNMAN-TYPE SIMULATORS

To study the time complexity of Schrödinger-Feynman-type simulators [2–4], *i.e.*, those that partition the circuit in sub-circuits and perform a full wave-function evolution of the sub-circuits for all the paths defined by the partition, we use the "qubit complexity" [3] for a given depth defined as follows.

For a bipartition, sub-circuits A and B are generated; if sub-circuit A has n_A qubits, B has n_B qubits, and there are α_{AB} CZ gates cut, then sub-circuits A and B have to be simulated $2^{\alpha_{AB}}$ times, with complexity 2^{n_A} and 2^{n_B} each, respectively, for the computation of one amplitude, or batch of amplitudes at a small overhead cost. The qubit complexity is in this case $\log_2 [2^{\alpha_{AB}} (2^{n_A} + 2^{n_B})]$.

For a partition in three regions, sub-circuits A, B, and C are generated, with n_A, n_B, and n_C qubits, respectively. The number of CZ gates cut is now α_{AB} between A and B, α_{AC} between A and C, and α_{BC} between B and C. In a naive computation, the qubit complexity would

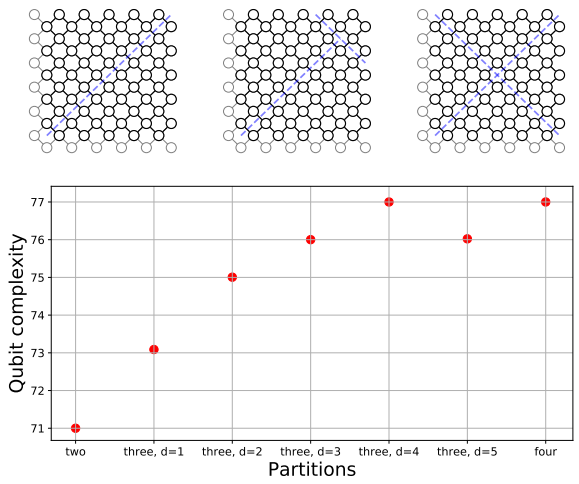

Figure C4. Qubit complexity of the simulation of Bristlecone-60 with depth (1+32+1) with Schrödinger-Feynman simulators [2–4] for different partition schemes. *Top:* From left to right, partition schemes considered for two partitions, three partitions, and four partitions; in the three partition case, d is the number of cuts avoided in the diagonal towards the top-right region, which for the example plotted is $d = 2$, and is extended to $d = 1 \ldots 5$ in our study. *Bottom:* Qubit complexity for the two-partition, three-partition, and four-partition schemes; we can see that the complexity is best in the two-partition case.

be $\log_2 \left[(2^{\alpha_{AB}} 2^{\alpha_{AC}} 2^{\alpha_{BC}}) \, (2^{n_A} + 2^{n_B} + 2^{n_C}) \right]$, since each of the three sub-circuits has to be simulated for each of the $2^{\alpha_{AB}} 2^{\alpha_{AC}} 2^{\alpha_{BC}}$ paths. However, it is possible to decrease the complexity by simulating sub-circuit A only once for each particular choice of path over cuts between A and B and cuts between A and C, and iterate (within this choice) over all paths over cuts between B and C. In that case, the cost is $\log_2 \left\{ (2^{\alpha_{AB}+\alpha_{AC}}) \left[2^{n_A} + 2^{\alpha_{BC}} (2^{n_B} + 2^{n_C}) \right] \right\}$. The qubit complexity is in this case minimized when A is the biggest sub-circuit, which can always be assumed without loss of generality.

For a partition in four sub-circuits (A, B, C, and D), where sub-circuits share CZ gates only between A and B, B and C, C and D, and between D and A, as in Fig. C4, the naive computation has qubit complexity $\log_2 \left[(2^{\alpha_{AB}+\alpha_{BC}+\alpha_{CD}+\alpha_{AD}}) \, \beta_{ABCD} \right]$, where $\beta_{ABCD} \equiv 2^{n_A} + 2^{n_B} + 2^{n_C} + 2^{n_D}$. However, it is possible, for each combination of paths between A and B, and C and D, to iterate over paths between B and C, and those between A and D, lowering the qubit complexity to $\log_2 \left[2^{\alpha_{AB}+\alpha_{CD}} \, (2^{\alpha_{BC}} \beta_{BC} + 2^{\alpha_{AD}} \beta_{AD}) \right]$, where $\beta_{BC} \equiv 2^{n_B} + 2^{n_C}$ and $\beta_{AD} \equiv 2^{n_A} + 2^{n_D}$.

We can see in Fig. C4 that for Bristlecone-60 with depth (1+32+1) the best performance is achieved for a partition into two sub-circuits, as is the case for the rectangular grids considered in Refs. [3, 4]. For a square grid $8 \times 8 \times (1 + 32 + 1)$, the qubit complexity is 65, which is lower than the best complexity found in this section for Bristlecone-60 with depth (1+32+1), *i.e.*, 71, even though the 8×8 square grid has four more qubits. This suggests that hard Bristlecone sub-lattices are harder to simulate than square (or rectangular) grids of the same (or smaller) number of qubits. Similar arguments apply to Bristlecone-70.

For the square grid $7 \times 7 \times (1 + 40 + 1)$ split into two sub-circuits [4], the qubit complexity is slightly over 63, and for the $7 \times 8 \times (1 + 40 + 1)$, the qubit complexity is 64.

D. SUPPLEMENTARY DETAILS ON HARDWARE USAGE AND RUNTIMES

Here we discuss the split in runtimes as reported in the main text (see Results), as well as the estimation of core-hours necessary for the simulations:

Split in runtimes. Investigation showed that the split in runtimes discussed in the main text (see Results) appears to be due to differing amounts of contention on the two sockets within a node (all the Pleiades and Electra nodes are dual-socket). We enforced the rule that all the jobs running on a particular node had the same number of threads. This could lead to there being a few unused cores. Individual threads were "pinned" to run on individual cores to avoid interference. However, the pinning strategy caused the unused cores to always be the lowest numbered cores (which are all on the first socket), and so fewer jobs ran concurrently on the first socket, causing them to see less contention, and therefore slightly higher performance than jobs that ran on the second socket. Unfavorable thread counts and core counts could also lead to one job per node having it's threads split across the two sockets; this creates yet another source of anomalous timings.

Core hour estimation. In our simulations on the Skylake nodes of Electra we used $P = 2304 \times 40 = 92160$ cores. Note that we consider 40 cores per node, even though we use only 39 in practice for the $7 \times 7 \times (1 + 40 + 1)$ simulations and 36 in the $8 \times 8 \times (1 + 40 + 1)$ simulations; this is due to the ability of modern Intel processors to "up-clock" their CPUs in favorable conditions (known as Dynamic Frequency Scaling), thus achieving a performance similar to the case where there are no idle cores. Similar estimates apply to the benchark of the other types of nodes used. Regarding our comparisons with Alibaba [5],

we take into account that the authors report the usage of $P = 2^{17} = 131072$ for their benchmark.

[1] Available at https://data.nas.nasa.gov/quail/data.php?dir=/quaildata/quantum/qcSim.

[2] S. Aaronson and L. Chen, in *LIPIcs-Leibniz International Proceedings in Informatics*, Vol. 79 (Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016).

[3] Z. Y. Chen, Q. Zhou, C. Xue, X. Yang, G. C. Guo, and G. P. Guo, Science Bulletin **63**, 964 (2018).

[4] I. L. Markov, A. Fatima, S. V. Isakov, and S. Boixo, arXiv:1807.10749 [quant-ph] (2018).

[5] J. Chen, F. Zhang, C. Huang, M. Newman, and Y. Shi, arXiv:1805.01450 [quant-ph] (2018).