

# Redefining the Quantum Supremacy Baseline With a New Generation Sunway Supercomputer

Xin Liu,<sup>1</sup> Chu Guo,<sup>2,\*</sup> Yong Liu,<sup>1,†</sup> Yuling Yang,<sup>1</sup> Jiawei Song,<sup>1</sup> Jie Gao,<sup>1</sup> Zhen Wang,<sup>1</sup> Wenzhao Wu,<sup>1</sup> Dajia Peng,<sup>3</sup> Pengpeng Zhao,<sup>1</sup> Fang Li,<sup>1,‡</sup> He-Liang Huang,<sup>2,4</sup> Haohuan Fu,<sup>3,§</sup> and Dexun Chen<sup>1</sup>

<sup>1</sup>National Supercomputing Center in Wuxi, Wuxi, Jiangsu, China

<sup>2</sup>Henan Key Laboratory of Quantum Information and Cryptography, Zhengzhou, Henan 450000, China

<sup>3</sup>Department of Earth System Science, Ministry of Education Key Laboratory for Earth System Modeling, Institute for Global Change Studies, Tsinghua University, Beijing 100084, China

<sup>4</sup>Shanghai Branch, CAS Centre for Excellence and Synergetic Innovation Centre in Quantum Information and Quantum Physics, University of Science and Technology of China, Hefei, Anhui 201315, China

(Dated: November 2, 2021)

A major milestone in the era of noisy intermediate scale quantum computers is *quantum supremacy* [Nature **574**, 505 (2019)] claimed on the Sycamore quantum processor of 53 qubits, which can perform a random circuit sampling task within 200 seconds while the same task is estimated to require a runtime of 10,000 years on Summit. This record has been renewed with two recent experiments on the Zuchongzhi 2.0 (56 qubits) and Zuchongzhi 2.1 (60 qubits) quantum processors. On the other front of quantum supremacy comparison, there has also been continuous improvements on both the classical simulation algorithm as well as the underlying hardware. And a fair justification of the computational advantages for those quantum supremacy experiments would require to practically simulate the same problems on current top supercomputers, which is still in lack. Here we report the full-scale simulations of these problems on new generation Sunway supercomputer, based on a customized tensor network contraction algorithm. Our benchmark shows that the most challenging sampling task performed on Sycamore can be accomplished within 1 week, thus collapsing the quantum supremacy claim of Sycamore. Additionally, we show that the XEB fidelities of the *quantum supremacy circuits* with up to 14 cycles can be verified in minutes, which also provides strong consistency check for quantum supremacy experiments. Our results redefine quantum supremacy baseline using the new generation Sunway supercomputer.

## INTRODUCTION

Ever since initially proposed in 1982 [1], quantum computers have long held the belief to be able to efficiently solve certain computational problems that are intractable for classical computers. After 40 years of theoretical and experimental developments [2–9], it has now come to the era of noisy intermediate scale quantum computers which can manipulate several tens of noisy physical qubits [10]. A major experimental milestone achieved along this way is the *quantum supremacy* experiment conducted with the 53-qubit superconducting quantum processor, entitled as Sycamore, by Google in 2019 [11], which demonstrates that for the specific task of sampling from a random quantum circuit, Sycamore can be  $10^9$  times faster than the best classical supercomputer Summit. Recently this record has been renewed with the 56-qubit and 60-qubit quantum processors entitled as Zuchongzhi 2.0 and Zuchongzhi 2.1 respectively [12, 13]. The rapid evolution of quantum processors also enables quantum algorithms on a larger scale [14–19].

Quantum supremacy is a competition between the best quantum and classical computers and it is a continuous instead of a single-shot effort. Pushing this competition to its extreme would be beneficial for both fields. In fact, both the classical simulation algorithms as well as the classical computing hardware have been upgrading rapidly along with the development of quantum computers. On the classical simulation algorithm side, a work from Alibaba in 2020 proposes a slicing and subtree reconfiguration scheme to search for near optimal tensor contract orders under a fixed memory bound [20]. When used in combination with tensor network

contraction (TNC) algorithm, they estimate that the most difficult sampling task on Sycamore, namely sampling from a random quantum circuit of 20 cycles (referred as Sycamore-20 afterwards), would only cost 19 days on Summit [20]. Instead of simulating exactly the same sampling task as performed on Sycamore, Ref. [21] shows that 2 million correlated exact amplitudes for Sycamore-20 can be computed with a 60-GPU cluster in 5 days, based on a customized big-head TNC algorithm. A similar strategy is also adapted and implemented on the new Sunway supercomputer with the runtime shortened to 304 seconds [22]. On the classical computing hardware side, highly efficient accelerators such as GPU and TPU have been upgrading rapidly and exascale computing systems are emerging. As an example, the new-generation NVIDIA A100 GPU has a 19.5-TFLOPS single-precision performance and a 312-TFLOPS half-precision performance.

At the time of writing, a full-scale simulation of the most difficult sampling tasks performed on those quantum processors, which integrates both novelties on the classical side, namely a state of the art TNC algorithm and a top supercomputer in the world, has not been reported. In this work, we report a highly efficient and full-scale implementation of a customized TNC algorithm on the new generation Sunway supercomputer. We demonstrate that the runtime to generate a perfect sample for Sycamore-20 is 440 seconds with single-precision arithmetic and 276 seconds with mixed-precision arithmetic. As a result, the most difficult sampling task performed on Sycamore, namely sampling 1 million bitstrings with 0.2% fidelity can be accomplished in 1 week, collapsing quantum supremacy claimed for Sycamore. Our results thus provide the *quantum supremacy baseline* for future bench-

mark.

Additionally, we verify three quantum supremacy circuits (defined as random quantum circuits with 12 cycles or above), namely Sycamore-12, Sycamore-14 and Zuchongzhi 2.0-12, by computing the exact amplitudes for 1 million experimentally generated bitstrings and then computing the cross benchmarking (XEB) fidelities for those circuits. The obtained values for XEB fidelities are slightly lower, but within errorbar, than those values estimated from the simplified variants used in the quantum supremacy experiments. Our result makes possible the real-time verification of large-scale quantum supremacy circuits which could become an indispensable tool for future development of quantum processors.

### EFFICIENT IMPLEMENTATION OF A CUSTOMIZED TNC ALGORITHM

Mathematically, an  $n$ -qubit quantum state is represented as a rank- $n$  tensor, with  $2^n$  entries of complex numbers, while single-qubit and two-qubit quantum gate operations can be represented as rank-2 and rank-4 tensors respectively. Applying a gate operation amounts to contracting the common tensor indices between the gate operation and the quantum state. This is the so-called Schrödinger algorithm with a time complexity  $O(m2^n)$  and space complexity  $O(2^n)$  for a quantum circuit with  $m$  gate operations. For Sycamore-20, we have  $n = 53$  and  $m \approx 400$  (single-qubit gate are neglected since they can be absorbed into two-qubit gates) and thus the time complexity is  $O(10^{18})$ . Ideally, this could be accomplished within seconds on an exascale supercomputer! The real limitation is the memory cost to store the quantum state, which is about 68 petabytes if stored as single-precision complex numbers and is at least one order of magnitude larger than the memory size of state of the art supercomputers. The implementation of the Schrödinger algorithm up to now is thus limited within 45 qubits [23–26].

To simulate the Sycamore quantum processor or beyond, an algorithm to trade time for space has to be used. Here we note the work from IBM researchers that leverages secondary storage to overcome the memory issue, which however is a more of a theoretical proposal instead of actual implementation [27]. During the past two years the method of choice to simulate the Sycamore-like quantum processor has gradually converged to a specific type of TNC algorithm [20, 21, 28], which treats the gate operations, the initial quantum state as well as the target computational basis (which are both made of separable rank-1 tensors) as a whole tensor network. Contracting this tensor network results in the exact amplitude for this basis. Such tensor network contraction is in general an NP-hard problem, whose performance greatly relies on a properly devised tensor contraction order. TNC is often used in conjunction with a *tensor index slicing* scheme, which slices a number of tensor indices such that the original tensor network is equivalent to the summation of a bunch of smaller tensor networks. This technique is important in

practice since the largest intermediate tensor during the contraction of the original tensor network could easily exceed the available amount of memory for large problems. As a result, the complexity of the original tensor network is that of each sliced tensor network times the total number of slices. Researchers from Alibaba propose an intertwined tensor slicing and subtree reconfiguration scheme to optimize the slicing and the tensor contraction order for the sliced tensor network altogether [20], whose strategy is also adopted in latest version of the package cotengra [29]. Currently, the tensor contraction order found for Sycamore-20 using different strategies are more or less of the same order of  $10^{18}$  [20, 21, 29] and in this work we use cotengra to produce a near-optimal tensor contraction order for later computation.

Given a specific tensor contraction order, the performance of TNC then depends mostly on the performance of pair-wise tensor contraction. The new generation Sunway supercomputer is powered by the homegrown SW26010P CPU, each with a total of 96 GB memory and is further divided into 6 core groups, each with 64 cores. The single-precision and half-precision performances are 14 TFLOPS and 53 TFLOPS respectively, and the memory bandwidth is 307 GB/s. Unfortunately, we find in practice that the resulting TNC produced by cotengra (also similar for other approaches) is dominated by highly skewed tensor contractions, namely contraction between a very high-rank tensor and a very low-rank tensor, for which the floating point arithmetic complexity is essentially of the same order as the memory access complexity (both proportional to the size of the larger tensor). As a result the ultimate performance of the tensor contraction is limited by the memory bandwidth. Nevertheless, to make the most utilization of our CPU architecture, we propose a fused tensor permutation and multiplication algorithm to push the efficiency of skewed tensor contraction to its extreme.

Our full-scale implementation starts with a two-level parallelization scheme which demonstrates to be most efficient for Sycamore-20: the inner level consists of 6 core groups (a single CPU) which takes a single sliced tensor network as input and distributes each tensor contraction onto the 384 cores; the outer level distributes all the slices into different CPUs. The tensor index slicing is done with a maximally allowed intermediate tensor size of  $2^{31}$  for single precision, which results in  $2^{22}$  slices, and  $2^{30}$  for mixed precision, which results in  $2^{23}$  slices. For the highly skewed pair-wise tensor contraction performed inside each CPU between a high-rank tensor  $A$  and a low-rank tensor  $B$ , we first store a copy of  $B$  in the local data memory (LDM) of each CPU core (256 KB), and then we fetch the slices of  $A$  into LDMs of each cores and perform a local matrix multiplication in parallel. In case  $B$  does not fit into LDM (which only happens for the Zuchongzhi series),  $B$  is further sliced and each LDM stores a single slice, then the different cores interchange their slices using remote memory access (RMA). The central design principle is that  $A$  and  $B$  are only loaded into the LDMs once. The algorithm is also shown in Fig. 1(b, c) (see Supplementary for more details). In practice we find that for typical tensor networks gen-

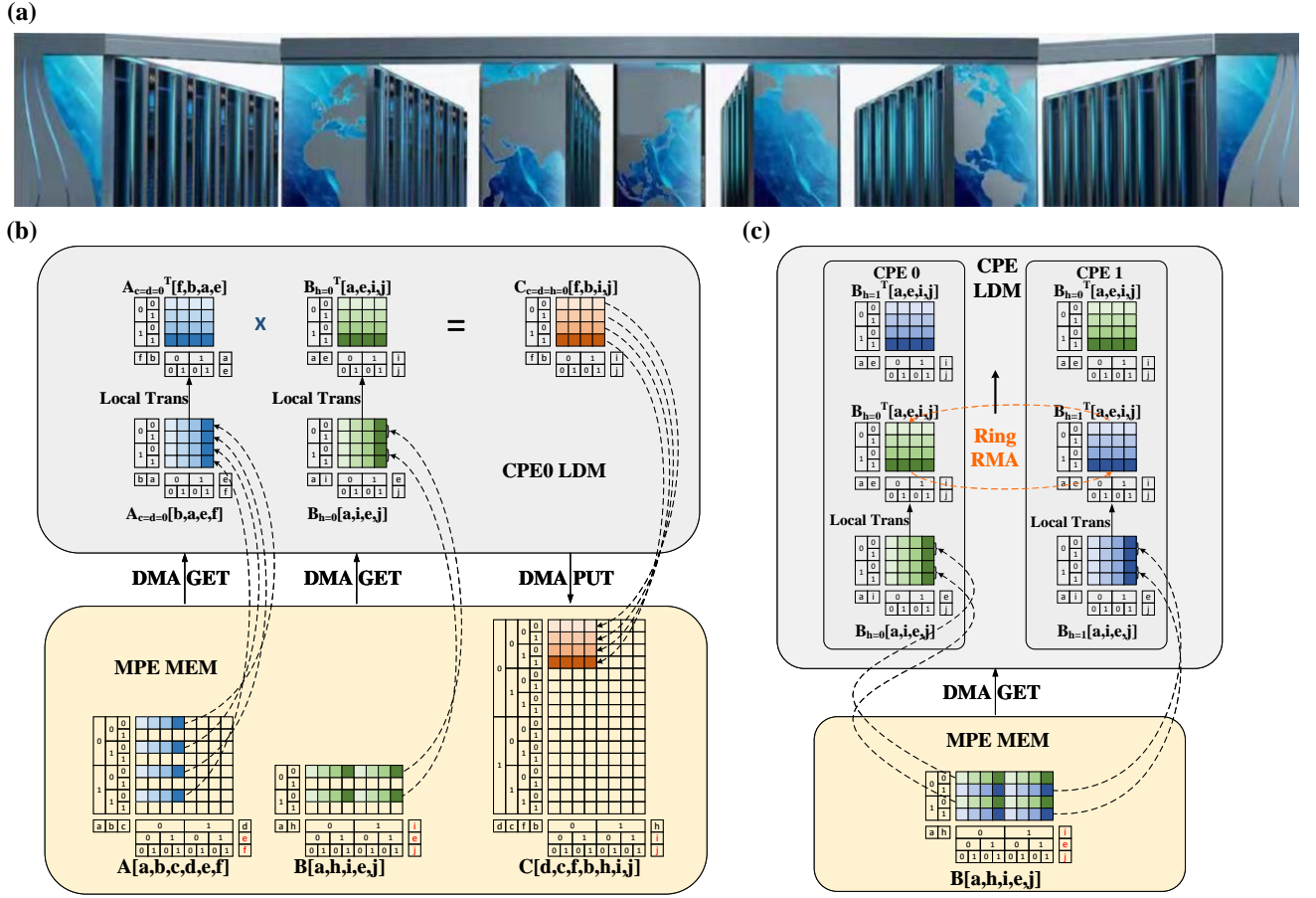


FIG. 1. (a) A rendered photo of the new generation Sunway supercomputer. (b) Demonstration of the fused tensor permutation and multiplication algorithm between a high-rank tensor  $A$  and a low-rank tensor  $B$ . (c) Demonstration of ring RMA when  $B$  is too large to fit into the LDM of a single core.

erated for the quantum supremacy circuits, our implementation can reach a 0.63 TFLOPS single-precision efficiency for each CPU (which surpasses the memory bandwidth by a factor of 2 due to the existences of a few computational intensive tensor contractions). Compared to the commonly used package jax [30], there is an average speedup of more than 25x and a maximum speedup of more than 100x. The near optimal software implementation together with the unprecedented parallelization scale over 41,932,800 cores allow us to tackle those quantum supremacy circuits which are believed to be extremely difficult or impossible to simulate previously.

To this end we note that the TNC algorithm used here is in sharp comparison with the ones used in Ref. [31–34], where the tensors in the time direction are pre-contracted, resulting in a two-dimensional tensor network with the same geometry as the quantum processor. In the latter case it has been shown that the FLOPS efficiency could easily exceed 60% [22]. However for Sycamore-20, each tensor in the resulting two-dimensional tensor network is too large to be stored on a single CPU, making it impractical for quantum supremacy circuits with more than 14 cycles [11].

## ESTABLISHING QUANTUM SUPREMACY BASELINE

The classical runtime of the quantum supremacy circuits are estimated by evaluating the runtime for generating one perfect sample (except in the case of Zuchongzhi 2.1 where sampling a single bitstring classically already becomes demanding). This is done by computing a batch of 64 amplitudes in a single run by leaving 6 qubits open and then perform frugal sampling, which could guarantee to produce one sample with probability close to 1 [20, 32]. Here we note that for TNC algorithm there is a very economic way to compute a correlated bunch of amplitudes in a single run by reusing the contraction outcome of a major portion of the tensor network, and iterate over the rest small portion. In this way, the overhead of computing a small bunch of correlated amplitudes is negligible compared to computing a single amplitude. Moreover, it has been shown that to match the sampling outcomes from a noisy quantum computer with an XEB fidelity  $f$ , the classical simulation cost can be simply reduced by a factor of  $f$  [35]. As a result, the classical complexity of generating 1 million samples with 0.2% XEB fidelity is equivalent to generating 2000 perfect samples.

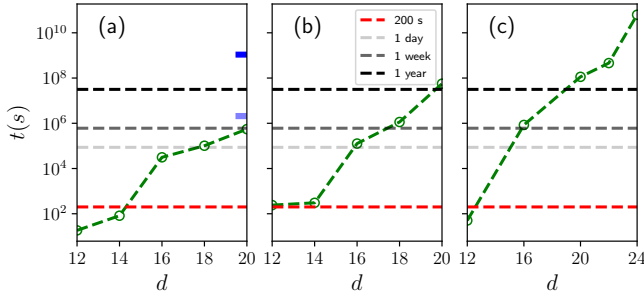


FIG. 2. Total runtime (green dashed line with circle) to simulate the same sampling task (producing 1 million bitstrings with the same XEB fidelity as the quantum processor) for quantum supremacy circuits on (a) Sycamore (b) Zuchongzhi 2.0 and (c) Zuchongzhi 2.1, based on extrapolation from the runtime to generate a single perfect sample. The x-axis  $d$  denotes the number of cycles. For Sycamore and Zuchongzhi 2.0, we directly generate one perfect sample. While for Zuchongzhi 2.1, we estimate the time to generate a perfect sample from the runtime to compute a single slice using a single CPU. The blue rectangle marks the total runtime for Ref. [21] on 60 GPUs while the light blue rectangle marks the estimated total runtime for Alibaba [20] on Summit.

The estimated runtimes for the Sycamore supremacy circuits are shown in Fig. 2(a). In particular, for Sycamore-20 our runtime for generating a perfect sample is 276 seconds, and thus generating 2000 perfect samples (or equivalently 1 million samples with 0.2% fidelity) would only require 6.4 days, which could be straightforwardly simulated with the new generation Sunway supercomputer. This is the fastest record till now, which clearly collapses the claim of quantum supremacy for the Sycamore processor. Moreover, the runtimes for Sycamore-12 and Sycamore-14 are only 18 and 82 seconds respectively (their estimated values in Ref. [11] are 2 hours and 2 weeks respectively), which are even faster than Sycamore (200 seconds).

In Fig. 2(b,c) we also show our estimated runtimes for Zuchongzhi 2.0 and Zuchongzhi 2.1 respectively. We can see that Zuchongzhi 2.0-20 and Zuchongzhi 2.1 with beyond 20 cycles require runtimes of more than 1 year, which are currently beyond our reach. The most difficult candidate is Zuchongzhi 2.1-24, for which generating a single perfect sample would require a classical runtime of around 5 years for us.

## VERIFYING QUANTUM SUPREMACY CIRCUITS

The derivation of the XEB fidelities for the quantum supremacy circuits is slightly subtle since to obtain the XEB fidelities one needs to compute amplitudes of the experimentally generated bitstrings with classical computers, which is considered not possible for those circuits. In fact, the estimated runtimes for computing 1 million exact amplitudes for Sycamore-12 and Sycamore-14 are 6 days and 4.26 years respectively [11]. The XEB fidelities of those quantum supremacy circuits are then estimated based on two differ-

ent approaches: 1) extrapolation based on component level fidelities which assumes a good suppression of the cross talks between gate operations and 2) extrapolation based on simplified variants for which a small portion of the gate operations is removed. The elided circuits are used for the latter purpose. Taking the elided version of Sycamore-20 for example, the number of removed gate operations is less than 30, which is small compared to the total number of gate operations which is more than 1000. However, the amount of entanglement in the underlying quantum states is reduced by more than half after removing those gates (thus the classical simulation complexity decreases exponentially). The equivalence between the XEB fidelities of the elided circuits and the corresponding supremacy circuits thus may not be easily justified.

Nevertheless, the XEB fidelity is a vital ingredient to characterize those near-term quantum processors. In the first place, the quantitative comparison between quantum and classical computers is only well-defined once the XEB fidelity of the quantum sampling process is precisely known. With our highly efficient random quantum circuit simulator on the new generation Sunway supercomputer, it is possible to directly compute a large number of exact amplitudes for quantum supremacy circuits with less than 14 cycles, with which we can compute the exact XEB fidelities and verify them against the estimated ones. Concretely, we compute a million exact amplitudes for Sycamore-12, Sycamore-14 and Zuchongzhi 2.0-12, from which we obtain their XEB fidelities as  $(1.34 \pm 0.1)\%$ ,  $(0.73 \pm 0.1)\%$  and  $(0.27 \pm 0.1)\%$  respectively, in comparison with the values of 1.4%, 0.9% and 0.37% as estimated from their elided versions. We can see that the exact XEB fidelities for those quantum supremacy circuits are slightly lower than their estimated values, but still within errorbar, which may be due to the reason that their estimated values are computed using a lot more bitstrings than 1 million.

We plot in Fig. 3 the histograms of those amplitudes and compare them to the theoretical probability density function (PDF) for the rescaled bitstring probability  $Np$  ( $N = 2^n$  and  $p$  is the probability) under the same XEB fidelity  $\mathcal{F}_{\text{XEB}}$  [11], namely

$$P_l(x|\mathcal{F}_{\text{XEB}}) = (\mathcal{F}_{\text{XEB}}x + (1 - \mathcal{F}_{\text{XEB}}))e^{-x}, \quad (1)$$

with  $x = Np$ . We can see that they agree well with each other, therefore the bitstrings generated with those quantum supremacy circuits indeed obey the Porter-Thomas distribution with corresponding XEB fidelities. This result provides a strong consistency check for those quantum supremacy experiments which could also be one of the major application scenarios for highly efficient classical simulators of quantum circuits in developing next-generation quantum processors.

## DISCUSSION

The motivation for developing efficient classical simulators for quantum circuits is ultimately driven by the lack of logi-



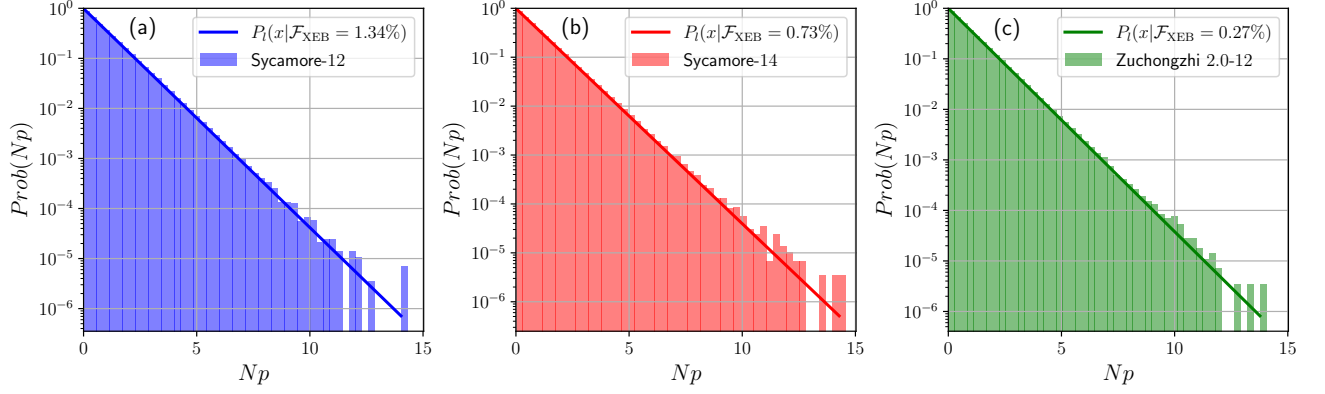


FIG. 3. Histograms for the distributions of the amplitudes corresponding to the experimentally generate bitstrings for (a) Sycamore-12, (b) Sycamore-14 and (c) Zuchongzhi 2.0-12. The solid lines denote the corresponding theoretically predictions with the same XEB fidelities as from Eq.(1).

cal qubits and the existence of cross talks in current quantum processors. The Sycamore and Zuchongzhi series of quantum processors reinforce the need to explore the classical computing capacity to its extreme, either for performance benchmarking or fidelity verification.

Although it is believed that the quantum processors will become exponentially more difficult to be simulated classically, our results together with other works during the last two years after Sycamore, seem to indicate that classical computational capacity for simulating random quantum circuits is growing in pace with the quantum counterparts. This is partially due to that the new Zuchongzhi series quantum processors grow more in terms of the number of qubits but less of the gate operation fidelities, and that the classical simulation algorithm, especially TNC based algorithms and classical accelerators are also progressing rapidly.

Our results herald a renewed starting of the quantum-classical competition in terms of simulating quantum supremacy experiments in exascale systems. There is still plenty of space for optimization on the classical side. In the first place, the memory bandwidth and the half-precision performance are apparently bottlenecks for our simulations, which are only 20% and 17% of that of NVIDIA A100 GPU. Since our current approach is essentially bounded by the memory bandwidth, we expect an immediate 5x performance increase using other CPU architectures with higher memory bandwidths. A more intelligently devised tensor contraction order could certainly help, especially if the balance between computation and memory could be better taken into account. Additionally, it is recently shown that to compute a large number of uncorrelated amplitudes it is still possible to reuse a large number of intermediate tensors and easily gain a 20x performance increase [36]. We thus believe that in near term the time to classically simulate Sycamore-20 would be reduced by 2 orders of magnitude (about 1.5 hours).

This work is partially supported by National Key R&D Program of China (2017YFA0604500), and National Natural Science Foundation of China (U1839206). C. G acknowledges

support from National Natural Science Foundation of China under Grants No. 11805279, No. 61833010, No. 12074117 and No. 12061131011. H.-L.H. acknowledges support from the Youth Talent Lifting Project (Grant No. 2020-JCJQ- QT-030), National Natural Science Foundation of China (Grant No. 11905294), China Postdoctoral Science Foundation, and the Open Research Fund from State Key Laboratory of High Performance Computing of China (Grant No. 201901-01).

---

\* [guochu604b@gmail.com](mailto:guochu604b@gmail.com)  
† [alexander.liu.99@163.com](mailto:alexander.liu.99@163.com)  
‡ [38349735@qq.com](mailto:38349735@qq.com)  
§ [haohuan@tsinghua.edu.cn](mailto:haohuan@tsinghua.edu.cn)

- [1] R. P. Feynman, International journal of theoretical physics **21**, 467 (1982).
- [2] P. W. Shor, in *Proceedings 35th annual symposium on foundations of computer science* (Ieee, 1994) pp. 124–134.
- [3] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver, Applied Physics Reviews **6**, 021318 (2019).
- [4] H.-L. Huang, D. Wu, D. Fan, and X. Zhu, Science China Information Sciences **63**, 180501 (2020).
- [5] S. Slussarenko and G. J. Pryde, Applied Physics Reviews **6**, 041303 (2019).
- [6] R. Blatt and C. F. Roos, Nature Physics **8**, 277 (2012).
- [7] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, Applied Physics Reviews **6**, 021314 (2019).
- [8] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature **549**, 195 (2017).
- [9] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, Reviews of Modern Physics **92**, 015003 (2020).
- [10] J. Preskill, Quantum **2**, 79 (2018).
- [11] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, Nature **574**, 505 (2019).
- [12] Y. Wu, W.-S. Bao, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan, *et al.*, Physical Review Letters **127**, 180501 (2021).
- [13] Q. Zhu, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan, M. Gong, *et al.*, Science Bulletin

- (2021).
- [14] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta, *Nature* **567**, 209 (2019).
  - [15] G. A. Quantum *et al.*, *Science (New York, NY)* **369**, 1084 (2020).
  - [16] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan, *Reviews of Modern Physics* **92**, 015003 (2020).
  - [17] M. P. Harrigan, K. J. Sung, M. Neeley, K. J. Satzinger, F. Arute, K. Arya, J. Atalaya, J. C. Bardin, R. Barends, S. Boixo, *et al.*, *Nature Physics* **17**, 332 (2021).
  - [18] V. Saggio, B. E. Asenbeck, A. Hamann, T. Strömberg, P. Schiavsky, V. Dunjko, N. Friis, N. C. Harris, M. Hochberg, D. Englund, *et al.*, *Nature* **591**, 229 (2021).
  - [19] X. Mi, M. Ippoliti, C. Quintana, A. Greene, Z. Chen, J. Gross, F. Arute, K. Arya, J. Atalaya, R. Babbush, *et al.*, arXiv preprint arXiv:2107.13571 (2021).
  - [20] C. Huang, F. Zhang, M. Newman, X. Ni, D. Ding, J. Cai, X. Gao, T. Wang, F. Wu, G. Zhang, *et al.*, *Nature Computational Science*, 1 (2021).
  - [21] F. Pan and P. Zhang, arXiv preprint arXiv:2103.03074 (2021).
  - [22] Y. Liu, X. Liu, F. Li, H. Fu, Y. Yang, J. Song, P. Zhao, Z. Wang, D. Peng, H. Chen, *et al.*, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2021) pp. 1–12.
  - [23] K. De Raedt, K. Michielsen, H. De Raedt, B. Trieu, G. Arnold, M. Richter, T. Lippert, H. Watanabe, and N. Ito, *Computer Physics Communications* **176**, 121 (2007).
  - [24] M. Smelyanskiy, N. P. Sawaya, and A. Aspuru-Guzik, arXiv:1601.07195 (2016).
  - [25] T. Häner and D. S. Steiger, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis* (2017) pp. 1–10.
  - [26] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, T. Magerlein, E. Solomonik, and R. Wisnieff, arXiv:1710.05867 (2017).
  - [27] E. Pednault, J. A. Gunnels, G. Nannicini, L. Horesh, and R. Wisnieff, arXiv preprint arXiv:1910.09534 (2019).
  - [28] I. L. Markov and Y. Shi, *SIAM Journal on Computing* **38**, 963 (2008).
  - [29] J. Gray and S. Kourtis, *Quantum* **5**, 410 (2021).
  - [30] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, “**JAX: composable transformations of Python+NumPy programs,**” (2018).
  - [31] C. Guo, Y. Liu, M. Xiong, S. Xue, X. Fu, A. Huang, X. Qiang, P. Xu, J. Liu, S. Zheng, *et al.*, *Physical Review Letters* **123**, 190501 (2019).
  - [32] B. Villalonga, S. Boixo, B. Nelson, C. Henze, E. Rieffel, R. Biswas, and S. Mandrà, *npj Quantum Information* **5**, 1 (2019).
  - [33] B. Villalonga, D. Lyakh, S. Boixo, H. Neven, T. S. Humble, R. Biswas, E. G. Rieffel, A. Ho, and S. Mandrà, *Quantum Science and Technology* **5**, 034003 (2020).
  - [34] C. Guo, Y. Zhao, and H.-L. Huang, *Physical Review Letters* **126**, 070502 (2021).
  - [35] I. L. Markov, A. Fatima, S. V. Isakov, and S. Boixo, arXiv:1807.10749 (2018).
  - [36] G. Kalachev, P. Pantelev, and M.-H. Yung, arXiv preprint arXiv:2108.05665 (2021).

# Supplementary Information: Redefining the Quantum Supremacy Baseline With a New Generation Sunway Supercomputer

(Dated: November 2, 2021)

## I. RANDOM QUANTUM CIRCUITS

The random quantum circuits (RQCs) on Sycamore and Zuchongzhi series of quantum processors share a similar structure which consists of interlacing layers of single-qubit gates and two-qubit gates, where each layer of two-qubit gates is counted as one cycle. The single-qubit gates are randomly and independently selected from the set  $\{\sqrt{X}, \sqrt{Y}, \sqrt{W}\}$  defined as

$$\sqrt{X} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & -i \\ -i & 1 \end{bmatrix}; \quad (1)$$

$$\sqrt{Y} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}; \quad (2)$$

$$\sqrt{W} = \frac{\sqrt{2}}{2} \begin{bmatrix} 1 & -\frac{\sqrt{2}}{2}(1+i) \\ \frac{\sqrt{2}}{2}(1-i) & 1 \end{bmatrix}. \quad (3)$$

The two-qubit gate is modeled by the 5-parameter fSim gate

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{i(\Delta_+ + \Delta_-)} \cos(\theta) & -ie^{i(\Delta_+ - \Delta_-, \text{off})} \sin(\theta) & 0 \\ 0 & -ie^{i(\Delta_+ + \Delta_-, \text{off})} \sin(\theta) & e^{i(\Delta_+ + \Delta_-)} \cos(\theta) & 0 \\ 0 & 0 & 0 & e^{i(2\Delta_+ - \phi)} \end{bmatrix}.$$

The values of those parameters are obtained through certain calibration procedures.

In all the three quantum supremacy experiments, 4 different patterns of two-qubit layers are used, which are denoted as  $A, B, C, D$ . Those 4 patterns are shown in Fig. 1, where each empty circle denotes one qubit and each line between two circles denotes a two-qubit gate operation. The shaded qubits and lines mean that those qubits as well as gate operations are not used in experiments, possibly due to high error rates. As a result each pair of nearest-neighbour qubits is operated on by a two-qubit gate once and only once in each 4 cycles containing all  $A, B, C, D$ . A RQC of a fixed number of cycles is then formed by repeating these 4 patterns, with randomly generated single-qubit gate layers in between. For example a RQC of 8 cycles is chosen as  $ABCD CDAB$ , and a RQC of 12 cycles is  $ABCD CDAB ABCD$ . RQCs with cycles which are not divided by 4 are implemented slightly differently for Sycamore and for Zuchongzhi series and we will not discuss about those details here. The overall design principle is to make the classical simulation cost as high as possible given the same number of qubits and the same amount of quantum gate operations.

In practice, the number of bitstrings generated experimentally should be larger than a minimal number to ensure that the uniform sampling with 0 XEB fidelity is rejected with certain level of confidence. This minimal number is thus related to the overall fidelity of the underlying RQC. As a result the number of samples required by the Zuchongzhi series experiments are larger than that required in the Sycamore experiments due to the lower fidelities, and that the number of samples should also be larger for RQCs with more cycles. However, here we simply choose a fixed number, that is, 1 million, for all those experiments

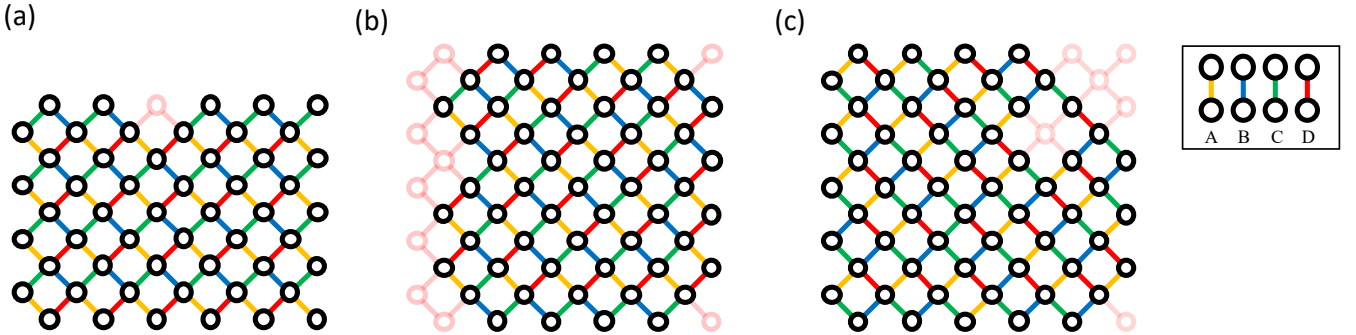


FIG. 1: Patterns of two-qubit gate layers for (a) Sycamore, (b) Zuchongzhi 2.0 and (c) Zuchongzhi 2.1.

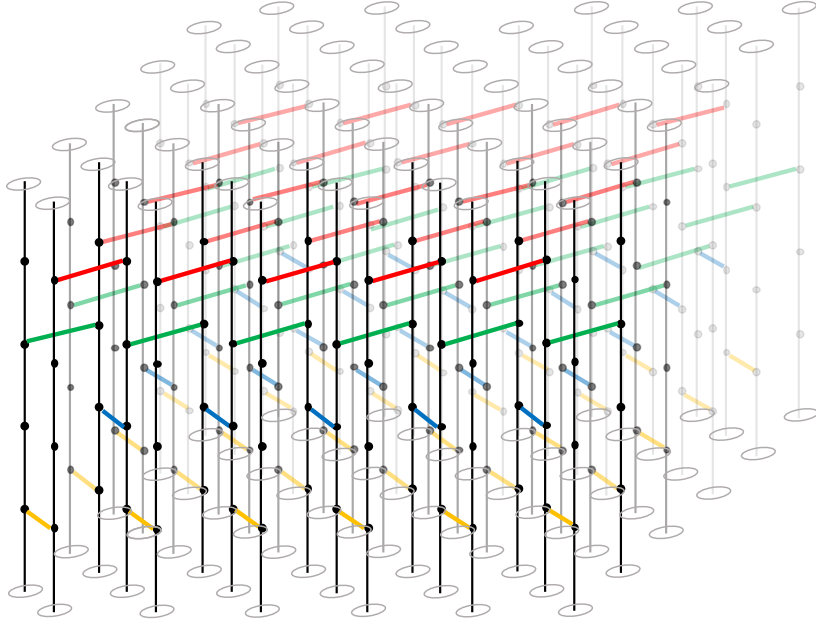


FIG. 2: Map of the task of computing a single amplitude into contracting a tensor network. The vertical direction denotes the time direction (from bottom up). The black lines separated by black balls denote tensor legs, each with dimension 2. The empty circles in the bottom and top layers represent the rank-1 tensors resulting from the initial state and final computational basis respectively. The black balls without any colored edges between them denote single-qubit gate operations. The colored edges together with two connected black balls denote two-qubit gate operations, where different colors denote two-qubit gate operations belonging to different patterns from  $A, B, C, D$ .

for a simple benchmarking since the complexity of sampling from the quantum processor and from our classical simulator (based on TNC algorithm) both grow linearly with the number of required samples (bitstrings).

## II. COMPONENT-LEVEL CALIBRATION

Ideally, one could calibrate each single-qubit gate and two-qubit gate and then the fidelity of a full RQC can be estimated as the product the fidelities of all the component gates. However due to the cross talks among different gate operations the fidelities of the RQCs estimated in this way would be lower than the actually measured values. Ref. [1] proposes a component-level calibration scheme to characterize the fidelities of the two-qubit gate operations, which is demonstrated to have better generalizability when used to predict fidelities of the full RQCs. This scheme is also adopted in Zuchongzhi 2.0 [2] and will be briefly reviewed in the following.

To characterize the two-qubit gate XEB fidelities for the two-qubit gate operations belonging to pattern  $A$ , for example, one use a circuit which only contains repeating layers of the  $A$  patterns (with single-qubit gate layers in between), and then measure all the XEB fidelities of these pairs of two qubits altogether. After subtracting the contributions of the single-qubit gate operations, whose XEB fidelities are characterized before hand, one obtains the XEB fidelities of all those two-qubit gate operations belonging to pattern  $A$  in a parallelized fashion. It turns out that the XEB fidelities obtained in this way for the two-qubit gate operations are slightly lower than those obtained from individual calibrations, but they can better predict the XEB fidelities of the full RQCs since the effects of cross talks have been absorbed into those fidelities to a certain level. In Ref. [3], however, an even more elaborated calibration scheme is used and we will not go into the details of it here. The main message here is that for current noisy quantum processors, it is difficult to completely get rid of the effects of cross talks. As a result, the full-circuit level benchmarking of the fidelities will be important, either as a brute-force calibration scheme itself, or as a consistency check for the more elaborated subsystem calibration schemes in the future.

## III. TENSOR NETWORK CONTRACTION ALGORITHM

Computing a single amplitude from the output of a quantum circuit can be naturally mapped to the problem of contracting a tensor network, as demonstrated in Fig. 2. The tensor network can also be viewed as a graph, which is assumed to be connected, since otherwise it means that the original RQC can be partitioned into two independent smaller quantum circuits,



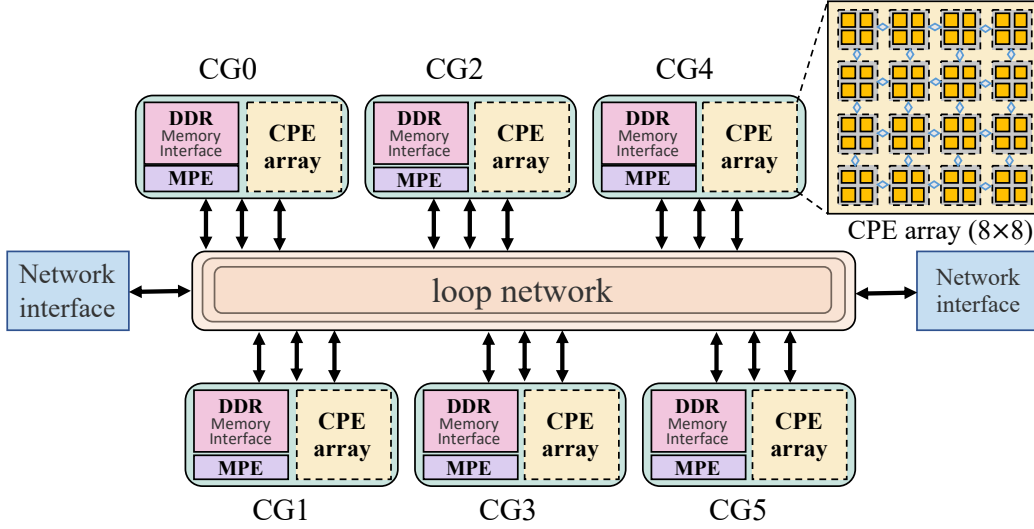


FIG. 3: Architecture of the SW26010P CPU.

which could then be simulated separately with much lower classical costs. Since rank-1 (resulting from the initial state and the final computation basis) and rank-2 tensors (single-qubit gates) can always be pre-absorbed into higher-rank tensors without increasing the sizes of tensors (two tensors with ranks larger than 2 will also be pre-contracted if the rank of resulting tensor is not larger than one of these two tensors), the tensor network will thus always be simplified to an equivalent one which only contains rank-3 tensors or above. As an example, for Sycamore-20 which contains a total of more than 430 two-qubit gate operations, the resulting tensor network contains less than 400 nodes, with each node a rank-4 tensor.

The complexity of contracting a tensor network is heavily dependent on the underlying tensor contract order, which is a list of two-tuples specifying the sequence of pair-wise tensor contractions. However, finding the optimal tensor contraction order for an arbitrary graph is an NP-hard problem and is almost impossible for our case with hundreds of nodes. Important progresses have been made in the last two years to find near optimal tensor contraction orders with heuristic algorithms [4–6]. In this work we combine a set of heuristic algorithms to produce a near optimal tensor contraction order for later processing. Concretely, we use the hypergraph partitioning algorithm from the package kahypar to decompose the initial tensor network into several connected subtrees, each of which is itself a smaller tensor network [7], and then we iteratively slice those subtrees and optimize each subtree (this is cheap and can be done by brute force since the number of nodes in the subtree is usually quite small) to reduce the the tensor network contraction complexity under a given memory bound for the largest intermediate tensor. We list the floating point arithmetic complexity corresponding to the tensor contraction orders found for several quantum supremacy circuits in TABLE I.

TABLE I: Floating point arithmetic complexities corresponding to the best tensor contraction orders found for several quantum supremacy circuits. Here the largest allow size for the intermediate tensors is  $2^{32}$ .

circuit	number of slices	complexity
Sycamore-20	$2^{21}$	$6.92 \times 10^{18}$
Zuchongzhi 2.0-20	$2^{29}$	$1.29 \times 10^{21}$
Zuchongzhi 2.1-20	$2^{29}$	$2.55 \times 10^{21}$
Zuchongzhi 2.1-24	$2^{40}$	$2.08 \times 10^{24}$

In practice, the tensor contraction order with a lower floating point arithmetic complexity may not be faster due to the memory intensive nature of the algorithm. To circumvent this problem, we generate 100 tensor contraction orders and run a single slice on a single CPU for each of them, and then choose the fastest one.

#### IV. CPU ARCHITECTURE AND THE TENSOR CONTRACTION ALGORITHMS

The SW26010P CPU contains 6 core groups (CGs). Each CG contains  $8 \times 8$  cores and a total of 16 GB of DDR4 memory with a bandwidth of 51.2 GB/s. Each core has an LDM of 256 KB. Data from the DDR4 memory can be loaded into the LDM of each core through the direct memory access (DMA), and different cores inside the same CG can interchange data through remote

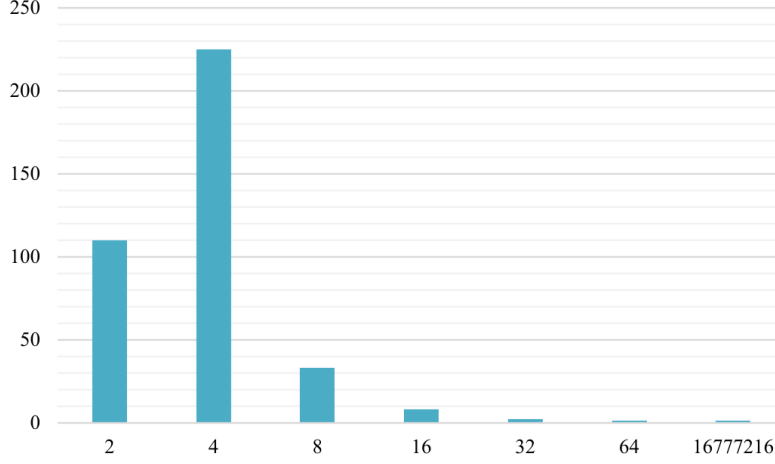


FIG. 4: Histogram of the contracted size  $k$  (x-axis) during the pair-wise contraction of a typical tensor network for Sycamore-20.

memory access (RMA). In total, a single SW26010P CPU has 384 cores and 96 GB memory, a 14 TFLOPS single-precision performance and a 53 TFLOPS half-precision performance, as well as a 307 GB/s memory bandwidth. The architecture of a single SW26010P CPU is shown in Fig. 3.

In our case the tensor network contraction is dominated by highly skewed pair-wise tensor contractions for which the memory access complexity is roughly equal to the floating point arithmetic complexity. However the memory bandwidth is only around 2.2% percent of the single-precision performance, as a result the performance of TNC is essentially bounded by the memory bandwidth and we have made an extreme effort to devise a fused tensor permutation and multiplication algorithm to make the best use of the limited memory bandwidth.

In the next we take the case of a pair-wise tensor contraction between a rank-7 tensor  $A[a, b, c, d, e, f, h]$  with a rank-4 tensor  $B[c, e, g, m]$  as an example, with the two common tensor indices  $c, e$  to be contracted. The result is a rank-7 tensor denoted as  $C[a, b, d, f, h, g, m]$ . A commonly used (and one of the most efficient) algorithm for tensor contraction is to permute the tensor indices of  $A$  and  $B$  and then perform matrix multiplication as

$$C[(a, b, d, f, h), (g, m)] = \sum_{(c, e)} A[(a, b, d, f, h), (c, e)] \times B[(c, e), (g, m)], \quad (4)$$

where  $(x, y, z)$  means to fuse the three individual tensor indices  $x, y, z$  into a single tensor index. For this algorithm, the permutation requires to load  $A$  and  $B$  once, which means to load the elements of them from the DDR4 memory into LDMs of each core, perform the permutation locally and then put the results back into DDR4 memory. After that, the following matrix multiplication would require to load the the permuted tensors once again. As a result, each element of  $A$  and  $B$  are loaded twice. This is ok if the resulting matrix multiplication is more or less balanced, that is, the first matrix has a shape  $p \times k$  and the second matrix has a shape  $k \times q$ , with  $p \approx k \approx q$ , since in this case the matrix multiplication complexity is  $O(k^3)$  while the memory access complexity is  $O(k^2)$ . However, in our case, as indicated in Eq.(4), we have  $p \gg k \approx q$  and thus both the matrix multiplication complexity and the memory access complexity is  $O(p)$ , taking into consideration that the memory bandwidth is much smaller than the floating point arithmetic performance, the complexity of the tensor multiplication would be dominated by the memory access. In Fig. 4, we plot the distribution of  $k$  appeared in the pair-wise tensor contractions corresponding to a particular tensor contraction order found for Sycamore-20.

Our fused tensor permutation and multiplication algorithm aims to reduce the memory access of  $A$  and  $B$  to only once. The main idea is as follows. First, since  $B$  is assumed to be very small, it is directed loaded into the LDMs of each cores through DMA. Then for a particular configuration of  $(a, b, d, f, h)$ , say  $(0, 1, 0, 1, 1)$ , we load  $A[0, 1, :, 0, :, 1, 1]$  into the LDM of a particular core through DMA, which is a vector of size 4 (or a matrix of size  $2 \times 2$ ). Then one can perform the tensor multiplication kernel locally which is simply a  $4 \times 4$  matrix times a vector of size 4, and the result is stored into  $C[0, 1, 0, 1, 1, :, :]$ . Different configurations of  $(a, b, d, f, h)$  can be loaded into different cores on the same CPU simultaneously with no data communication at all (the same for the local matrix multiplication kernel), thus can be done in parallel. In practice, we also aggregate the data transfer and the computation by combining  $(c, e)$  with  $(a, b, d)$ , namely each time we will fetch  $2^5$  elements of  $A$  into an LDM such that the 8 numbers corresponding to the first three indices  $(a, b, c)$  are fetched contiguously, and then the local matrix-vector multiplication becomes a matrix-matrix multiplication (a  $4 \times 4$  matrix times a  $4 \times 8$  matrix). The speedup of the tensor contraction algorithm compared to the commonly used library jax is shown in Fig. 5, where we select up to 10 best tensor



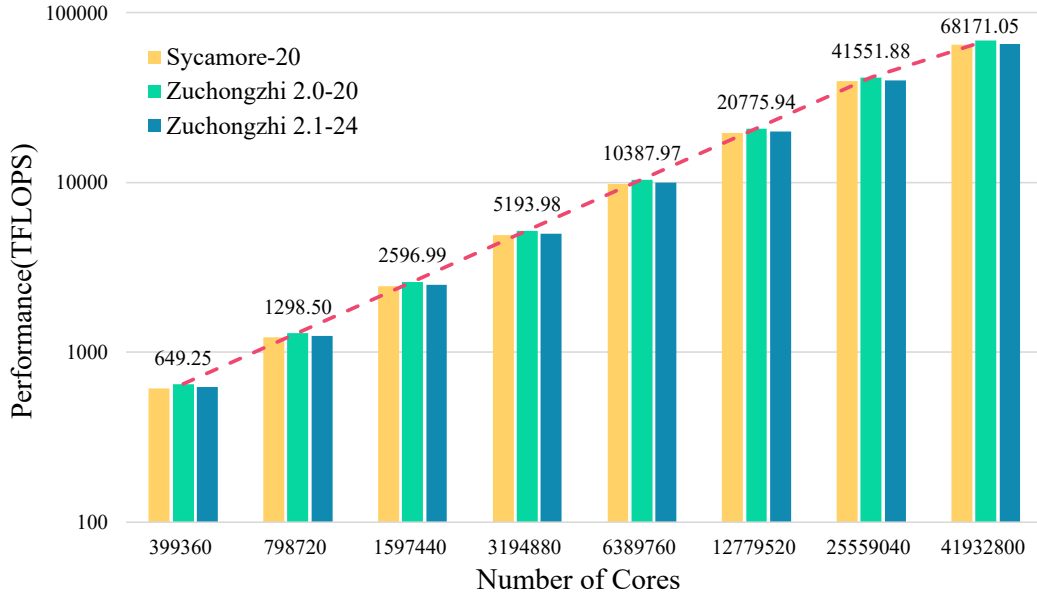


FIG. 6: Scaling of FLOPS performance against the number of cores used.

- 
- [1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell, *et al.*, *Nature* **574**, 505 (2019).
  - [2] Y. Wu, W.-S. Bao, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan, *et al.*, *Physical Review Letters* **127**, 180501 (2021).
  - [3] Q. Zhu, S. Cao, F. Chen, M.-C. Chen, X. Chen, T.-H. Chung, H. Deng, Y. Du, D. Fan, M. Gong, *et al.*, *Science Bulletin* (2021).
  - [4] J. Gray and S. Kourtis, *Quantum* **5**, 410 (2021).
  - [5] C. Huang, F. Zhang, M. Newman, X. Ni, D. Ding, J. Cai, X. Gao, T. Wang, F. Wu, G. Zhang, *et al.*, *Nature Computational Science* , 1 (2021).
  - [6] F. Pan and P. Zhang, arXiv preprint arXiv:2103.03074 (2021).
  - [7] S. Schlag, *High-Quality Hypergraph Partitioning*, Ph.D. thesis, Karlsruhe Institute of Technology, Germany (2020).
  - [8] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, “*JAX: composable transformations of Python+NumPy programs,*” (2018).