The Journal of Open Source Software

TNC: Distributed Tensor Network Contractions in Rust

Manuel Geiger [1], **Qunsheng Huang** [1], **and Christian B. Mendl** [1,2]

1 School for Computation, Information and Technology, Technical University of Munich, Germany **2** Institute for Advanced Study, Technical University of Munich, Germany

DOI: 10.21105/joss.09598

Software

- Review
- Repository
- Archive

Editor: Vincent Knight

Reviewers:

- @Luthaf
- @emapuljak

Submitted: 09 September 2025
Published: 02 June 2026

Summary

Since real quantum computers continue to have high error rates, classical error-free simulation of circuits is essential for verification, benchmarking, and algorithm design. A commonly used tool for the simulation is tensor networks, which represent a quantum circuit as network of multi-dimensional arrays that must be multiplied ("contracted") to compute the result of the circuit. TNC is a Rust-based library for efficiently contracting large tensor networks. Designed for distributed-memory environments common in high-performance computing (HPC), TNC partitions tensor networks to enable parallel contraction across multiple compute nodes. While optimized for quantum circuit simulation, TNC is general enough to contract arbitrary tensor networks, combining high performance with Rust's memory safety guarantees.

Statement of need

With the rapid advances in quantum computing, ever larger quantum circuits are being explored, which require more computing resources. Real quantum hardware is often not yet available, however, has limited qubit counts, and has high error rates due to imperfect gates and noise induced by the environment. Hence, research on efficient classical simulation methods is crucial to bridge this gap. Tensor networks have been shown to be a viable tool for classical simulation, even disproving two major claims of quantum advantage, i.e., the realization of an algorithm on real quantum hardware which would be infeasible to simulate on classical hardware (Pan et al., 2022; Patra et al., 2024). Apart from quantum simulation, they are also used in other areas, such as machine learning, quantum optimization, quantum chemistry, and quantum error correction (García & Romero, 2024).

However, existing libraries for tensor network contractions often only consider shared-memory parallelism (Pfeifer et al., 2015), are not open-source (Bayraktar et al., 2023), or are written in Python (Lykov et al., 2021) and hence less suited for HPC. Furthermore, parallelization is usually done by a technique called *slicing*, which incurs computational overhead. Popular libraries using slicing include ExaTN (Lyakh et al., 2022) and Jet (Vincent et al., 2022). TNC addresses these limitations by targeting distributed-memory systems, being fully open-source, implemented in Rust, and employing partitioning rather than slicing for parallelization. The use of Rust further provides strong guarantees of memory safety.

Features

The user can construct quantum circuits using the library's API or import them from code written in the widely-used OpenQASM2 language (Cross et al., 2017). From a given circuit, the user can then generate tensor networks that compute the expectation value, the amplitudes to specific measurement outcomes, or the full state vector. Furthermore, random tensor networks can be created as well, for instance, in the form of the original Sycamore experiment (Arute et

al., 2019). The library supports serialization of tensors and tensor networks to and from HDF5 files.

Tensors in TNC can be defined hierarchically, enabling tensor networks in which vertices may represent either raw tensors or nested subnetworks. This hierarchical representation naturally expresses partitioning-based parallelization and supports multiple parallelization levels.

For contraction, the library partitions tensor networks based on the number of available compute nodes. It obtains an initial partitioning using the hypergraph partitioning library KaHyPar (Andre et al., 2018). However, these initial partitionings are often not optimal in terms of contraction cost. Therefore, our library features different algorithms to improve an initial partitioning, such as greedy rebalancing, simulated annealing, and a genetic algorithm. As a cost function, we use a parallelism-aware computation cost metric. The final partitioning thus achieves better time-to-solution by better load balancing between compute nodes. In a recent publication, we showed that partitionings optimized with the simulated annealing method have contraction costs that are competitive with state-of-the-art methods (Geiger et al., 2025).

The library can find contraction paths, which specify the order of contraction operations, using different methods from the cotengra library (Gray & Kourtis, 2021). It then performs the individual contractions using matrix-matrix multiplications via MKL (Intel, n.d.) or a pure Rust backend, with tensor transpositions performed using the hptt library (Springer et al., 2017).

To execute distributed contractions, TNC performs the contractions of different partitions in parallel across compute nodes. Communication and data exchange between nodes are coordinated using MPI, enabling scalable execution on distributed-memory HPC systems.

Acknowledgements

We acknowledge the funding received for the MUNIQC-SC initiative under funding number 13N16191 from the VDI technology center as part of the German BMBF program. The work is also supported by the Bavarian state government via the BayQS project with funds from the Hightech Agenda Bayern, and via the Munich Quantum Valley, section K7, with funds from the Hightech Agenda Bayern Plus.

References

Andre, R., Schlag, S., & Schulz, C. (2018). Memetic multilevel hypergraph partitioning. *Proceedings of the Genetic and Evolutionary Computation Conference*, 347–354. https://doi.org/10.1145/3205455.3205475

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., Biswas, R., Boixo, S., Brandao, F. G. S. L., Buell, D. A., Burkett, B., Chen, Y., Chen, Z., Chiaro, B., Collins, R., Courtney, W., Dunsworth, A., Farhi, E., Foxen, B., … Martinis, J. M. (2019). Quantum supremacy using a programmable superconducting processor. *Nature*, *574*(7779), 505–510. https://doi.org/10.1038/s41586-019-1666-5

Bayraktar, H., Charara, A., Clark, D., Cohen, S., Costa, T., Fang, Y.-L. L., Gao, Y., Guan, J., Gunnels, J., Haidar, A., Hehn, A., Hohnerbach, M., Jones, M., Lubowe, T., Lyakh, D., Morino, S., Springer, P., Stanwyck, S., Terentyev, I., … Yamaguchi, T. (2023). cuQuantum SDK: A high-performance library for accelerating quantum science. *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, *01*, 1050–1061. https://doi.org/10.1109/QCE57702.2023.00119

Cross, A. W., Bishop, L. S., Smolin, J. A., & Gambetta, J. M. (2017). *Open quantum assembly language*. https://doi.org/10.48550/arxiv.1707.03429

García, M. D., & Romero, A. M. (2024). *Survey on computational applications of tensor network simulations*. arXiv. https://doi.org/10.48550/arxiv.2408.05011

Geiger, M., Huang, Q., & Mendl, C. B. (2025). *Optimizing tensor network partitioning using simulated annealing*. https://doi.org/10.48550/arxiv.2507.20667

Gray, J., & Kourtis, S. (2021). Hyper-optimized tensor network contraction. *Quantum*, *5*, 410. https://doi.org/10.22331/q-2021-03-15-410

Intel. (n.d.). *oneAPI math kernel library (oneMKL)*. https://software.intel.com/en-us/intel-mkl.

Lyakh, D. I., Nguyen, T., Claudino, D., Dumitrescu, E., & McCaskey, A. J. (2022). ExaTN: Scalable GPU-accelerated high-performance processing of general tensor networks at exascale. *Frontiers Appl. Math. Stat.*, *8*, 838601. https://doi.org/10.3389/fams.2022.838601

Lykov, D., Alexeev, Y., Ibrahim, C., & Galda, A. (2021). QTensor: Fast QAOA quantum simulator. *APS March Meeting Abstracts*, *2021*, M34.007.

Pan, F., Chen, K., & Zhang, P. (2022). Solving the sampling problem of the sycamore quantum circuits. *Physical Review Letters*, *129*, 090502. https://doi.org/10.1103/PhysRevLett.129.090502

Patra, S., Jahromi, S. S., Singh, S., & Orús, R. (2024). Efficient tensor network simulation of IBM's largest quantum processors. *Physical Review Letters*, *6*, 013326. https://doi.org/10.1103/PhysRevResearch.6.013326

Pfeifer, R. N. C., Evenbly, G., Singh, S., & Vidal, G. (2015). *NCON: A tensor network contractor for MATLAB*. https://arxiv.org/abs/1402.0939

Springer, P., Su, T., & Bientinesi, P. (2017). *HPTT: A high-performance tensor transposition C++ library*. arXiv. https://doi.org/10.48550/arxiv.1704.04374

Vincent, T., O'Riordan, L. J., Andrenkov, M., Brown, J., Killoran, N., Qi, H., & Dhand, I. (2022). Jet: Fast quantum circuit simulations with parallel task-based tensor-network contraction. *Quantum*, *6*, 709. https://doi.org/10.22331/q-2022-05-09-709