QMatchaTea

Accelerating Quantum Circuit Emulations with Tensor Networks on HPC Systems

Vito Palmisano[1,2], Marco Ballarin[3], Gabriella Bettonte[2], Daniel Jaschke[3,4], Elise Jennings[1], Sara Marzella[2], Simone Montangero[3], Daniele Ottaviani[2], Martin Ruefenacht[1]

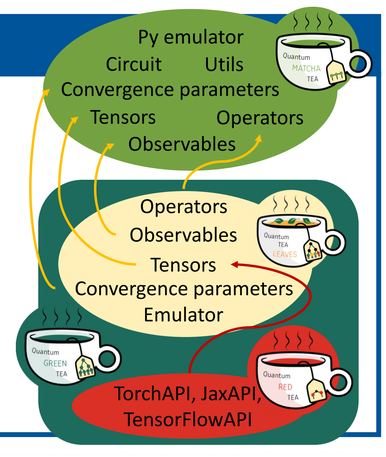
[1] ParTec AG, Germany; [2] SCAI - SuperComputing Applications and Innovation – CINECA, Italy; [3] University of Padova and INFN, Sezione di Padova, Italy; [4] Institute for complex quantum systems, Ulm University, Germany

Abstract

Quantum Matcha TEA (QMatchaTea) is a powerful quantum computer emulator within the *Quantum Tensor Network Emulator Applications (Quantum TEA)* suite, specifically designed to emulate quantum circuit using Matrix Product States (MPS) and Tree Tensor Networks (TTN), optimized for high-performance computing (HPC) environments. It is entirely written in Python, making it user-friendly and abstracting the need to manage backend operations.

QMatchaTea harnesses the efficiency of the tensor networks (TNs) methods, exploiting the sparsity and entanglement properties of quantum states to emulate large quantum systems with less memory resources compared to other emulation methods.

Moreover, the software integrates advanced linear algebra optimizations and parallel computing techniques, offering adaptability across various HPC architectures. QMatchaTea supports single-node emulation with GPU acceleration and distributed multi-node CPU emulation via Message Passing Interface (MPI). The tool's flexibility is further enhanced by support for multiple linear algebra backends, including NumPy/CuPy, PyTorch, JAX, and TensorFlow, allowing users to maximize the potential of their HPC infrastructure.



Scan the QR

https://baltig.infn.it/quantum_matcha_tea/py_api_quantum_matcha_tea

About QMatchaTea

Key Features

- **Tensor Network Methods**:
  - Uses MPS and TTN to emulate quantum systems.
  - Optimized memory efficiency.
- **Backends and Flexibility**: Offers a variety of tensor modules backends to optimize performance for different use cases:
  - *NumPy* and *Cupy* for CPU and GPU emulation.
  - Additional support for *PyTorch*, *JAX*, and *TensorFlow* for broader adaptability to HPC infrastructures [1].

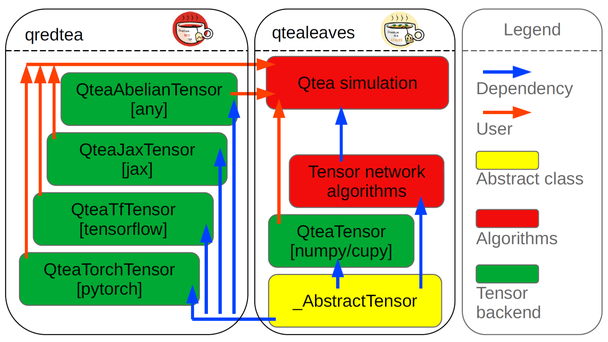


Figure 1. Tensor modules backends implementation [1]. The way they are implemented, allows to use any of them easily.

- **Scalability**: Scalable to HPC clusters using MPI, making it suitable for large-scale simulations.
- **Compatibility:** QmatchaTea integrates with *Qiskit* to define quantum circuits for simulation.

Matrix Product State Tensor Networks

- A **Tensor network (TN)** is a mathematical structure used to efficiently decomposing a large, global tensor into smaller, interrelated tensors, particularly in systems with many interacting components (e.g. quantum states) [2, 3, 4, 5].
- **Matrix Product State (MPS)** is one of the most widely used TN structures [2, 3, 4, 5]:
  - It represents a large quantum state as a **chain of small tensors**:
    - **each tensor corresponds to a qubit**;
    - the **edges between tensors represent the entanglement** between neighboring qubits.
  - To compress entanglement, **MPS performs Singular Value Decomposition (SVD) and truncates the bond dimension χ** [3, 5].
  - We can compute the Fidelity Lower Bound of the truncation.

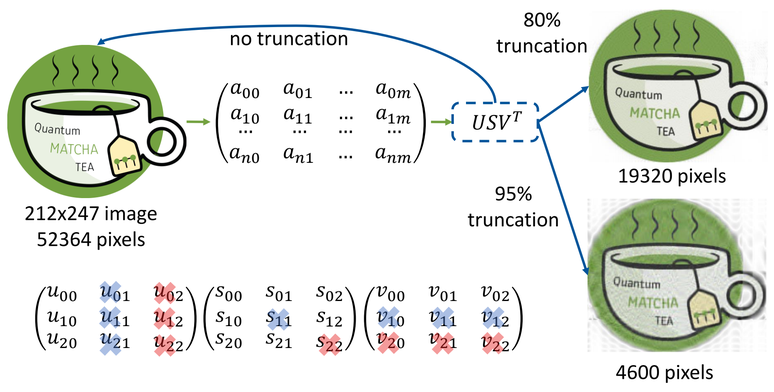


Figure 2. MPS compression of entanglement via SVD can be compared to image compression. The more we truncate the bond dimension χ, the more we increase the approximation (decrease the fidelity lower bound).

Takeaways:

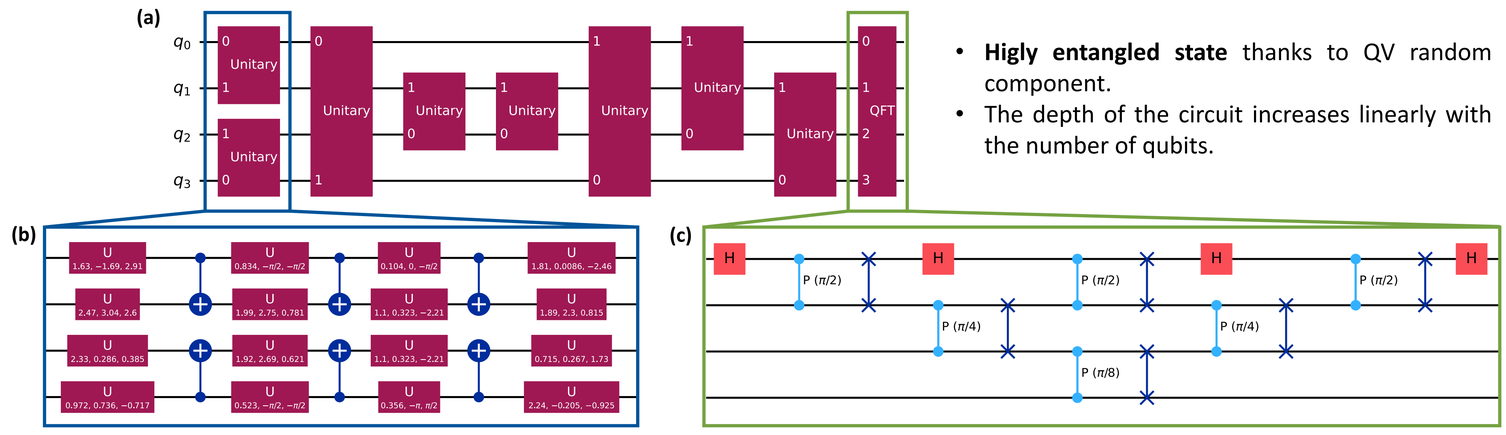
- TNs compress the quantum correlations between subsystems, i.e., **TNs compress entanglement** [4, 5].
- MPS simulations are **not limited by the number of qubits but by the entanglement** [4, 5].
- The memory requirement for qubits is reduced w.r.t. exact state emulation methods: $O(2^n) \longrightarrow O(2n\chi^2)$ [5].

Our emulation environment

LEONARDO	Booster partition	DCGP partition
Nodes	3456	1536
Processors	single socket 32 cores Intel Ice Lake CPU	dual socket 56 cores Intel Sapphire Rapids CPU
Accelerators	4 x NVIDIA Ampere GPUs/node, 64GB	-
Cores	32 cores/node	112 cores/node
RAM	512 (8x64) GB DDR4 3200 MHz	512 (16 x 32) GB DDR5 4800 MHz

Benchmarking results

Our emulated circuit: Quantum Volume (QV) random circuit + Quantum Fourier Transform (QFT)



- **Higly entangled state** thanks to QV random component.
- The depth of the circuit increases linearly with the number of qubits.

Figure 3. Each layer of the Quantum Volume (QV) circuit is a random permutation of the qubits followed by a randomly selected 2-qubit gate operating on pairs of qubits. In the emulated model circuit, the structure incorporates a number of layers d equal to the number of qubits n [6]. When n is an odd number, it results in one qubit remaining idle in every layer. The circuit in figure (a) depicts a QV circuit implemented using Qiskit having $n = d = 4$. In (b) the implementation of QV's first layer is showed; the implementation of the others layers follow a similar scheme. In (c) it is possible to see a parallelizable version of Quantum Fourier Transform (QFT) [5].

Benchmarking results from the European supercomputer Leonardo

Where not specified, the emulation results shown below were performed using the following **settings**: single CPU as device (Leonardo Booster partition), maximum bond dimension $\chi = 2^{\lfloor n/2 \rfloor}$, Numpy or Cupy as tensor module backend.

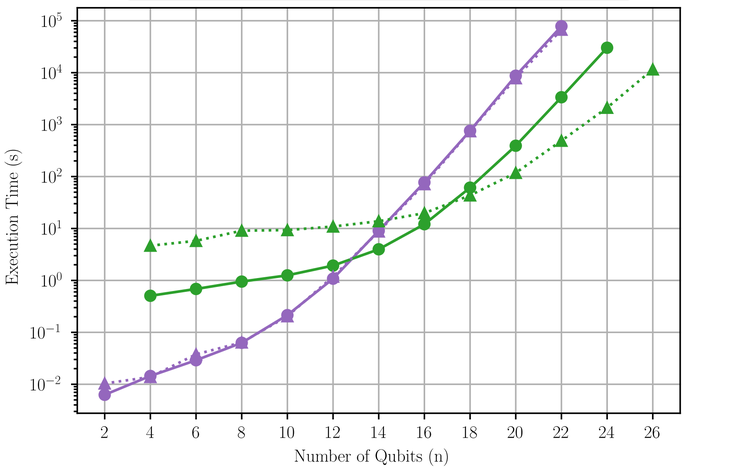


Figure 4. Comparison between different HPC settings: single thread, single GPU, 2 threads, 4 threads and 8 threads. The left y-axis reports QMatchaTea emulation times. Instead, the right y-axis reports the ratio between the execution times of the various HPC settings and the execution time of the single thread. It is possible to see that utilizing 2 threads effectively reduces the execution times, with a ratio between 0.8 and 0.6, but that using more than 2 threads does not comport any improvement. Notably, the scalability of multi-threading remains relatively constant from 16 qubits onward, whereas GPU scalability improves with the increasing number of qubits. For less than 18 qubits, transporting information to/from the GPU is more expensive than GPU emulation itself.

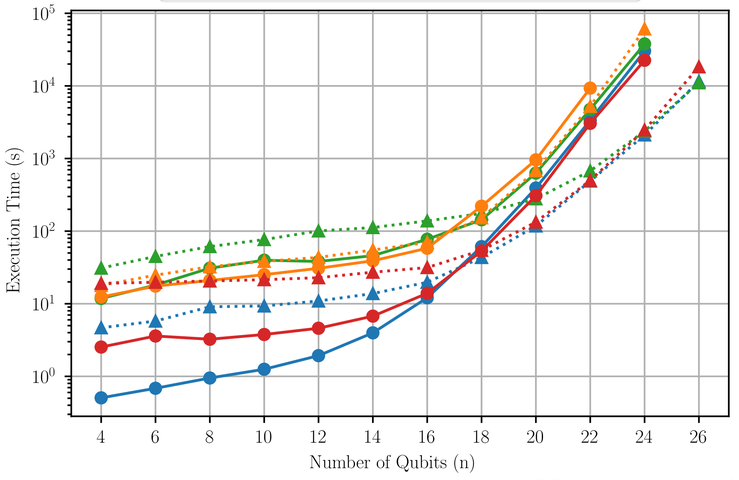


Figure 5. Comparison of five different MPS emulations having different maximum bond dimensions χ values. The comparison is made using a single CPU. The used χ values are functions of the number of qubits n; the five used functions are given in the legend. $\chi = 2^{\lfloor n/2 \rfloor}$ represents the maximum theoretical value that the bond dimension can reach during the emulation and is the case where truncation is not performed. Consequently, in this case we have the highest emulation times, but also a fidelity of 1. The more we truncate, the faster the emulation will be, but the fidelity LB will also decrease.

Figure 6. Execution times comparison between QMatchaTea and Qiskit. The comparison is made for two different HPC emulation settings, single CPU and single GPU. It is possible to see how QMatchaTea emulation is faster than Qiskit emulation concerning both the HPC settings when we emulate more than 14 qubits in the GPU case and from 14 qubits on in the single thread case.

Figure 7. Execution times comparison between the different tensor module backends available in QMatchaTea. The comparison is made for two different HPC emulation settings, single CPU and single GPU. Notably, for all the tensor module backends, 18 qubits is the turning point where GPU becomes faster than CPU. It is also possible to see how, on Leonardo, the best backends to use are Numpy/Cupy and Torch in both scenarios. On different HPC systems, results may vary.

More Info and Next Steps

- **QMatchaTea** is part of the whole quantum initiative at Cineca and QMatchaTea v1.1.4 is **available on Leonardo**:
  1. module load profile/quantum
  2. module load qmatcha_tea
- **Next steps**:
  - **Multi-GPU emulation implementation**.
  - Benchmarking of **Torch Intel's optimized version**.
  - Performances comparison with others MPS TNs emulators.

Bibliography

[1] D. Jaschke et. al, "Benchmarking Quantum Red TEA on CPUs, GPUs, and TPUs", 5 Sep 2024.
[2] Román Orús, "A practical introduction to tensor networks: Matrix product states and projected entangled pair states", Annals of Physics, vol. 349, Pages 117-158, 2014.
[3] P. Silvi et. al, "The Tensor Networks Anthology: Simulation techniques for many-body quantum lattice systems", SciPost Phys. Lect. Notes, 2019.
[4] L. Tagliacozzo et al., "Scaling of entanglement support for matrix product states", Physical Review B 78, jul 2008.
[5] M. Ballarin and S. Montangero, "Quantum Computer Simulation via Tensor Networks", Padua Thesis and Dissertation Archive, sep 2021.
[6] A. W. Cross et. al, "Validating quantum computers using randomized model circuits", Physical Review A, vol. 100, no. 3, sep 2019.