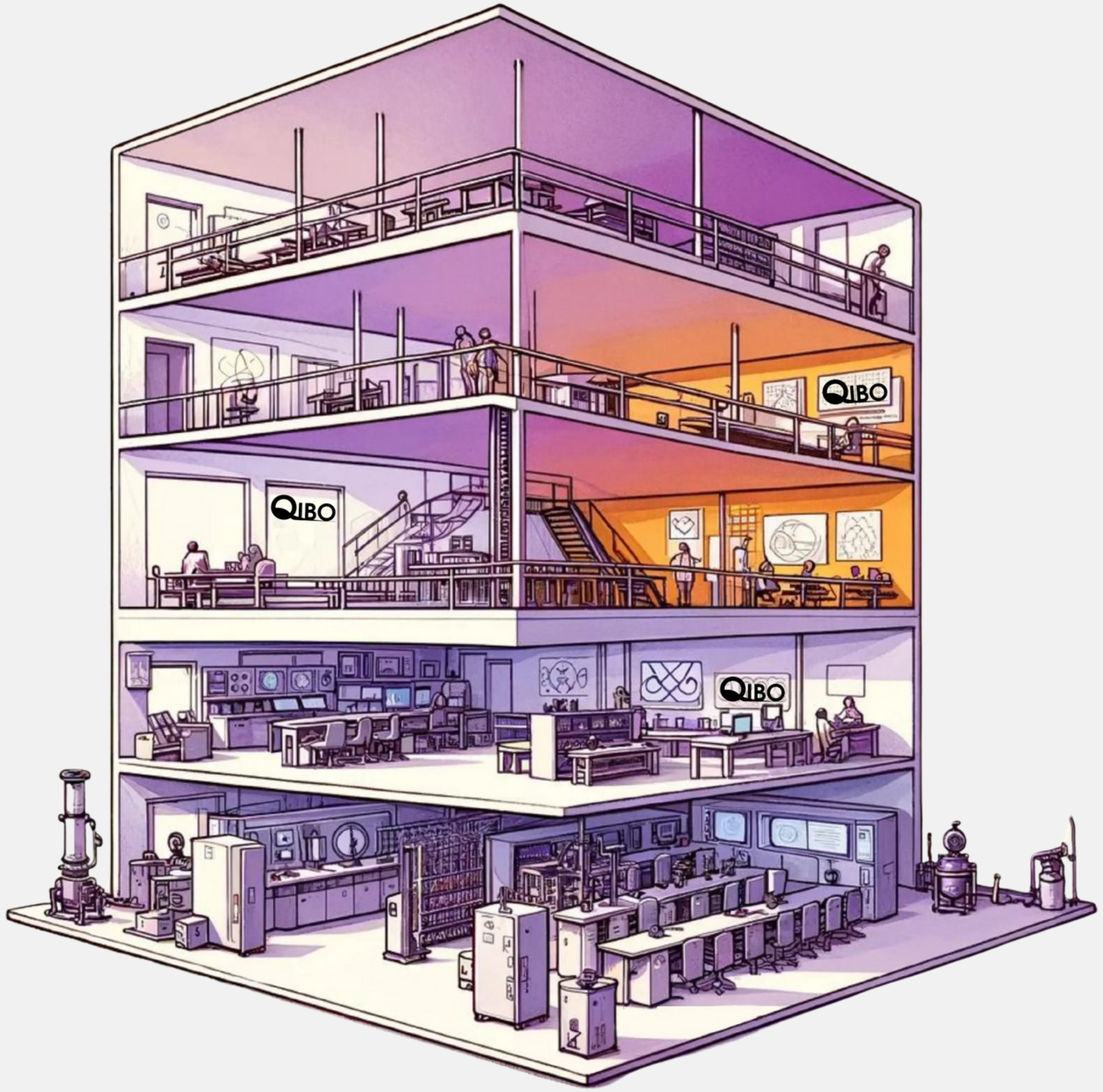


ACAT2024, Stony Brook, March 11th

Quantum simulation with just-in-time compilation

Andrea Pasquale, on the behalf of the QiboTeam

INFN Istituto Nazionale di Fisica Nucleare

UNIVERSITÀ DEGLI STUDI DI MILANO

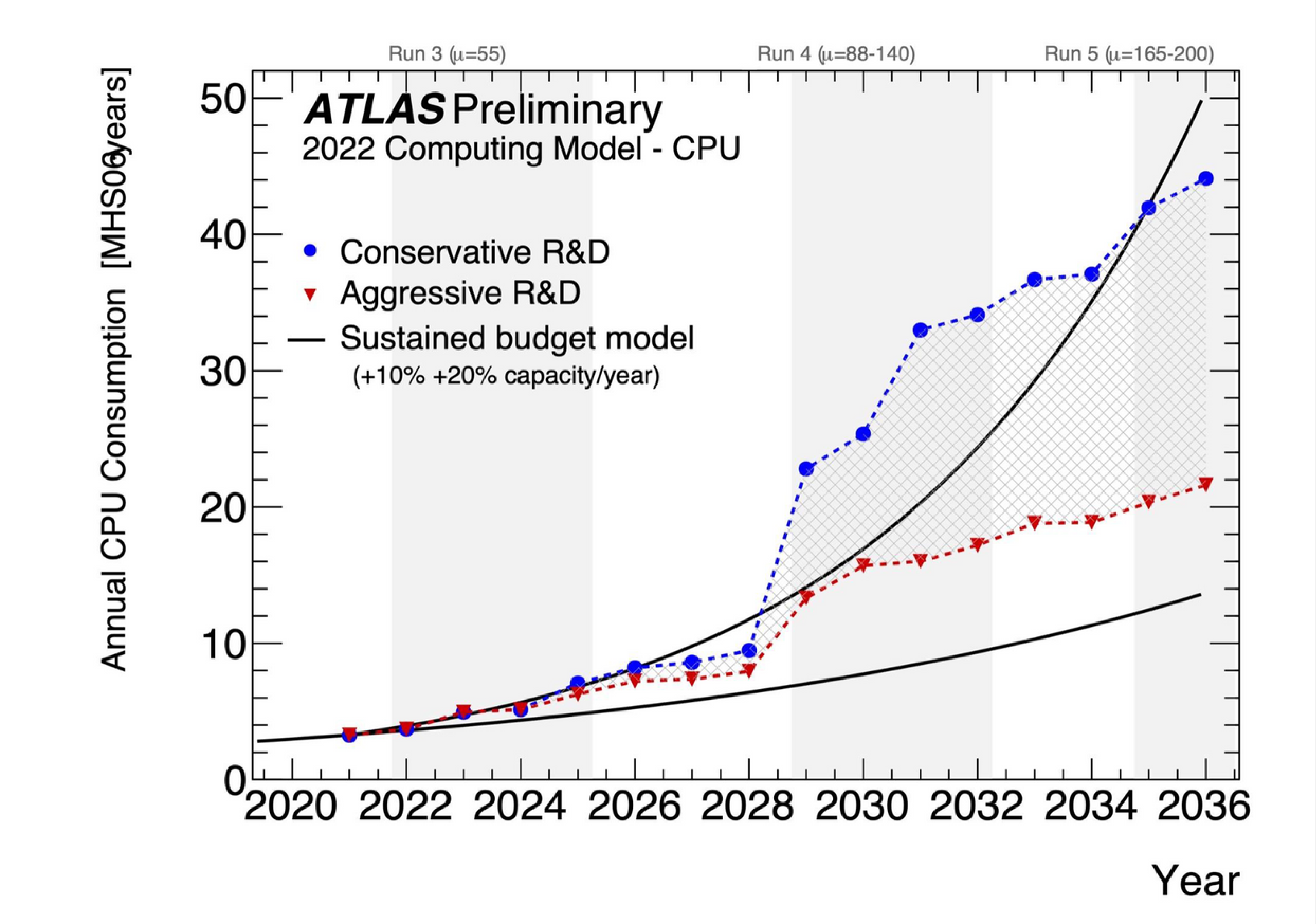
QIBO

TII Technology Innovation Institute

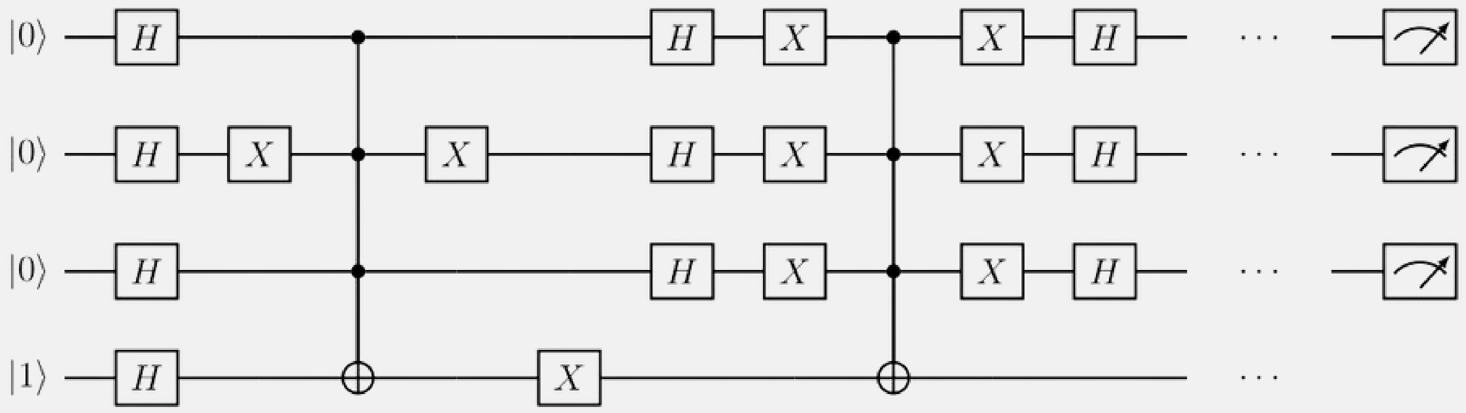
Why people in physics should care about quantum computing?

Someone is already trying...

- A. Perez et al, arXiv:2011.13934
- C. Bravo-Pietro et al, arXiv:2110.06933
- J. Cruz-Martinez et al, arXiv:2308.05657
- F. Rehm et al, arXiv:2307.05253
- M. Robbiati et al, arXiv:2311.05680



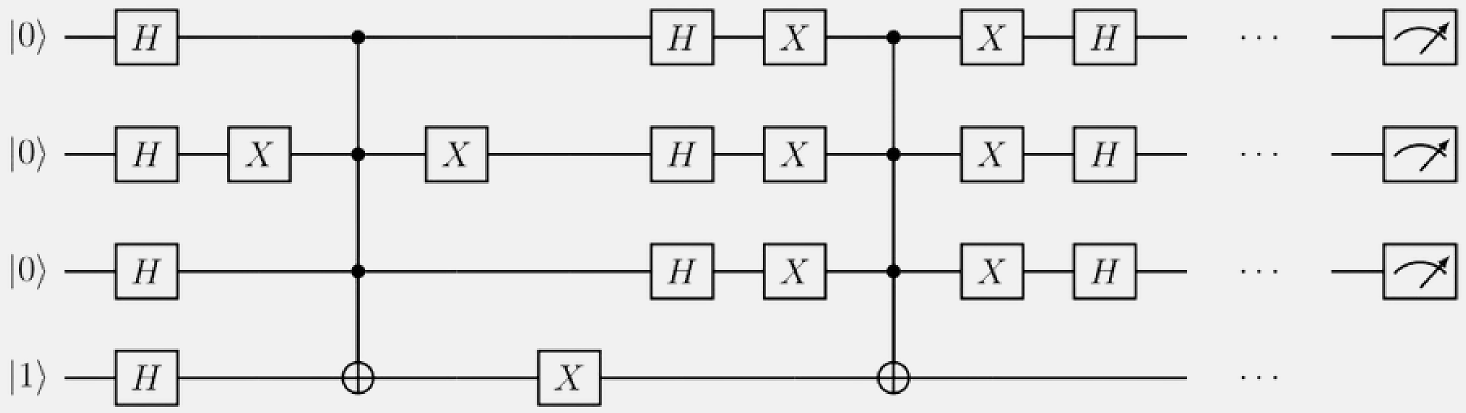
Introduction to Quantum Computing



Example of a Grover algorithm

Why do we care about simulating quantum computers?

Introduction to Quantum Computing

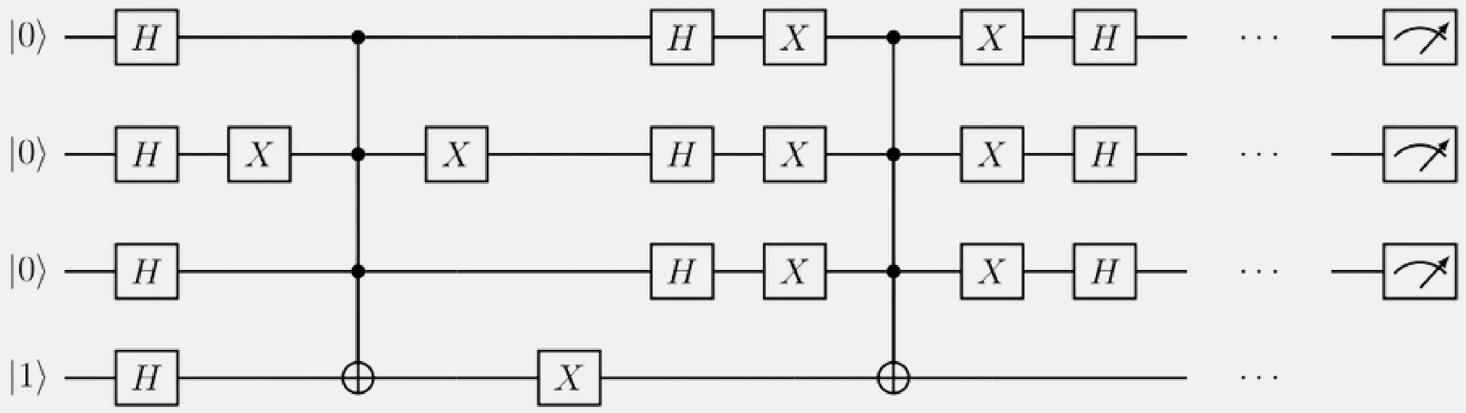


Example of a Grover algorithm

Why do we care about simulating quantum computers?

- Classical simulation is fundamental to understand and design quantum hardware
- To test and verify quantum algorithms before quantum hardware is ready

Introduction to Quantum Computing



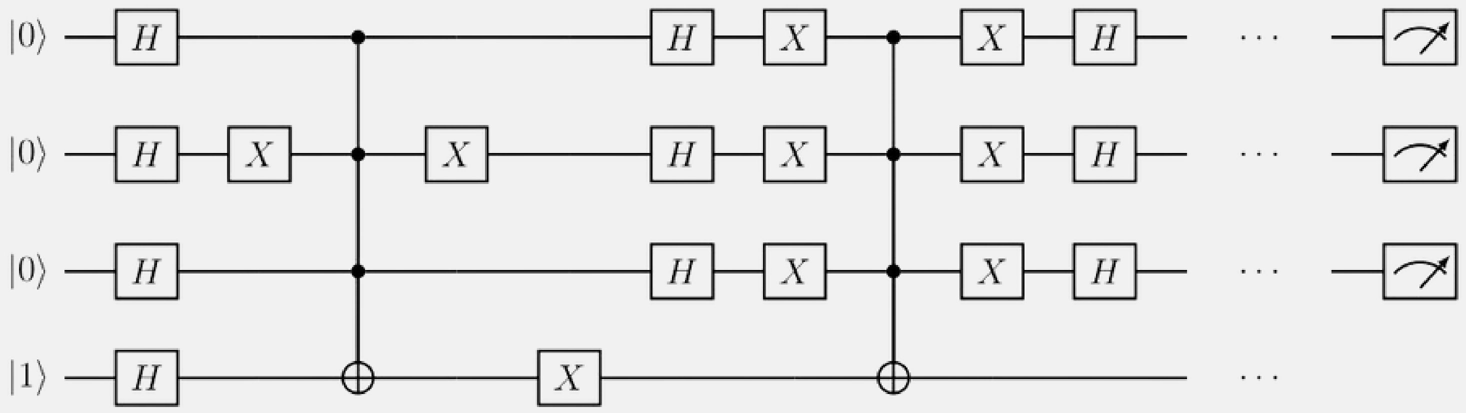
Example of a Grover algorithm

Why do we care about simulating quantum computers?

- Classical simulation is fundamental to understand and design quantum hardware
- To test and verify quantum algorithms before quantum hardware is ready

Simulating quantum computers is hard

Introduction to Quantum Computing



Example of a Grover algorithm

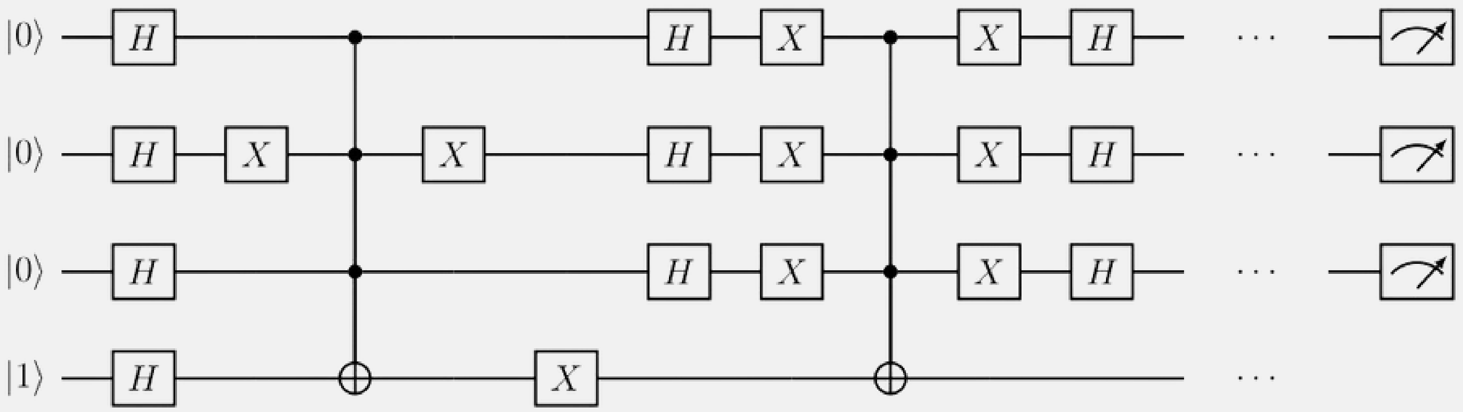
Why do we care about simulating quantum computers?

- Classical simulation is fundamental to understand and design quantum hardware
- To test and verify quantum algorithms before quantum hardware is ready

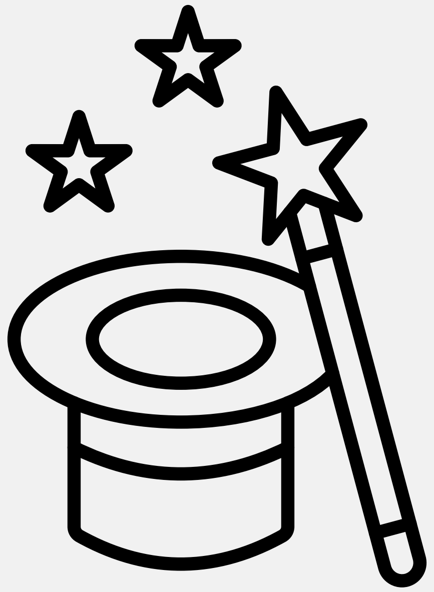
Simulating quantum computers is hard

- Memory management
- Operation needs to be optimized
- Needs to explore different architectures

Introduction to Quantum Computing



Example of a Grover algorithm



Therefore, we need a framework that has some “tricks” to efficiently perform quantum simulation

Why do we care about simulating quantum computers?

- Classical simulation is fundamental to understand and design quantum hardware
- To test and verify quantum algorithms before quantum hardware is ready

Simulating quantum computers is hard

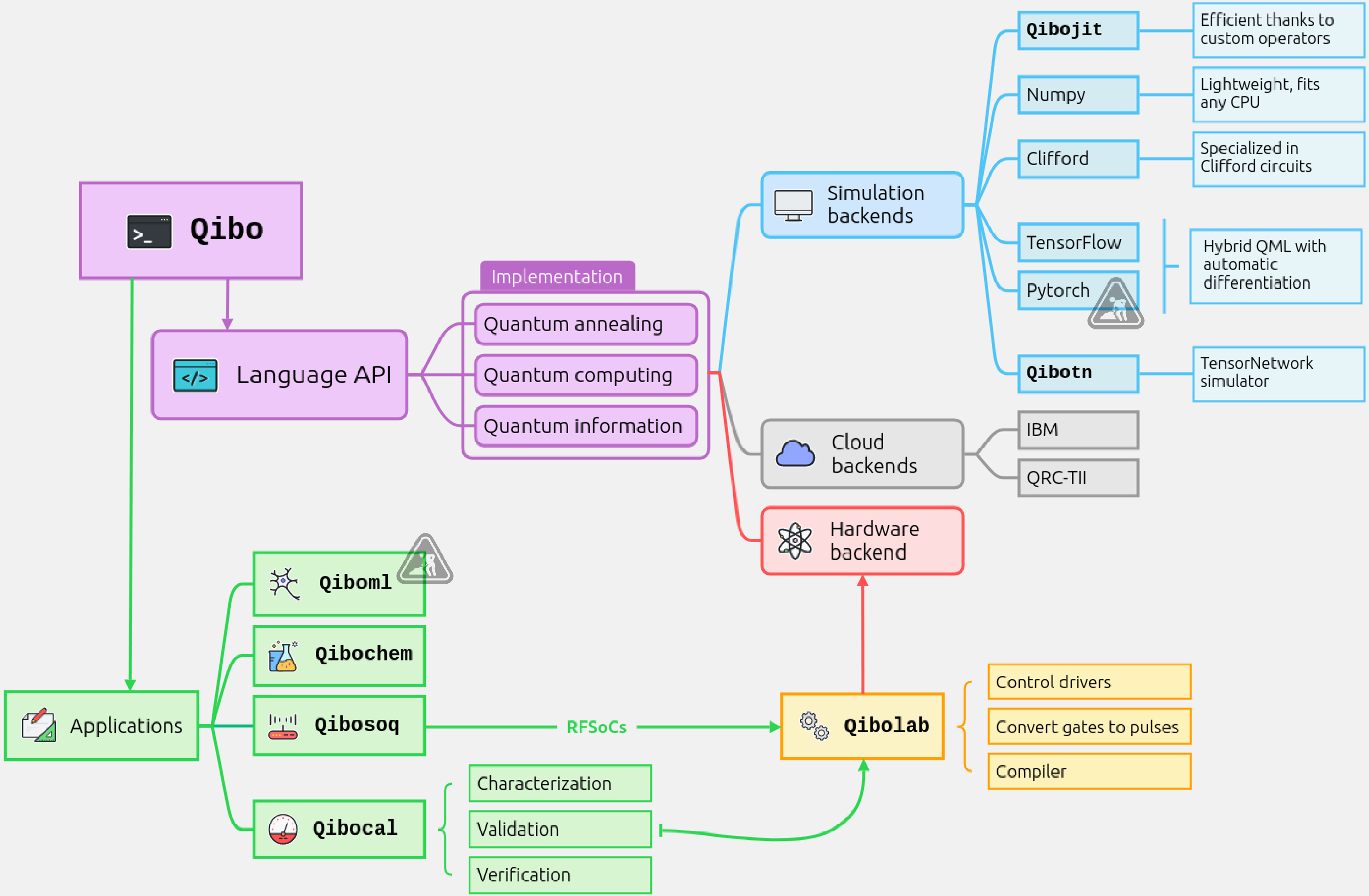
- Memory management
- Operation needs to be optimized
- Needs to explore different architectures

Qibo

A quantum computing framework for simulation and hardware execution

Stavros Efthymiou et al 2022 Quantum Sci. Technol. 7 015018

https://github.com/qiboteam/qibo



Qibo
Language API
Implementation
Quantum annealing
Quantum computing
Quantum information
Simulation backends
Qibojit
Efficient thanks to custom operators
Numpy
Lightweight, fits any CPU
Clifford
Specialized in Clifford circuits
TensorFlow
Pytorch
Hybrid QML with automatic differentiation
Qibotn
TensorNetwork simulator
Cloud backends
IBM
QRC-TII
Hardware backend
Applications
Qiboml
Qibochem
Qibosoq
RFSoCs
Qibocal
Characterization
Validation
Verification
Qibolab
Control drivers
Convert gates to pulses
Compiler

Qibojit

Full state vector simulation with just-time-compilation

https://github.com/qiboteam/qibojit

Full state vector simulation in a nutshell

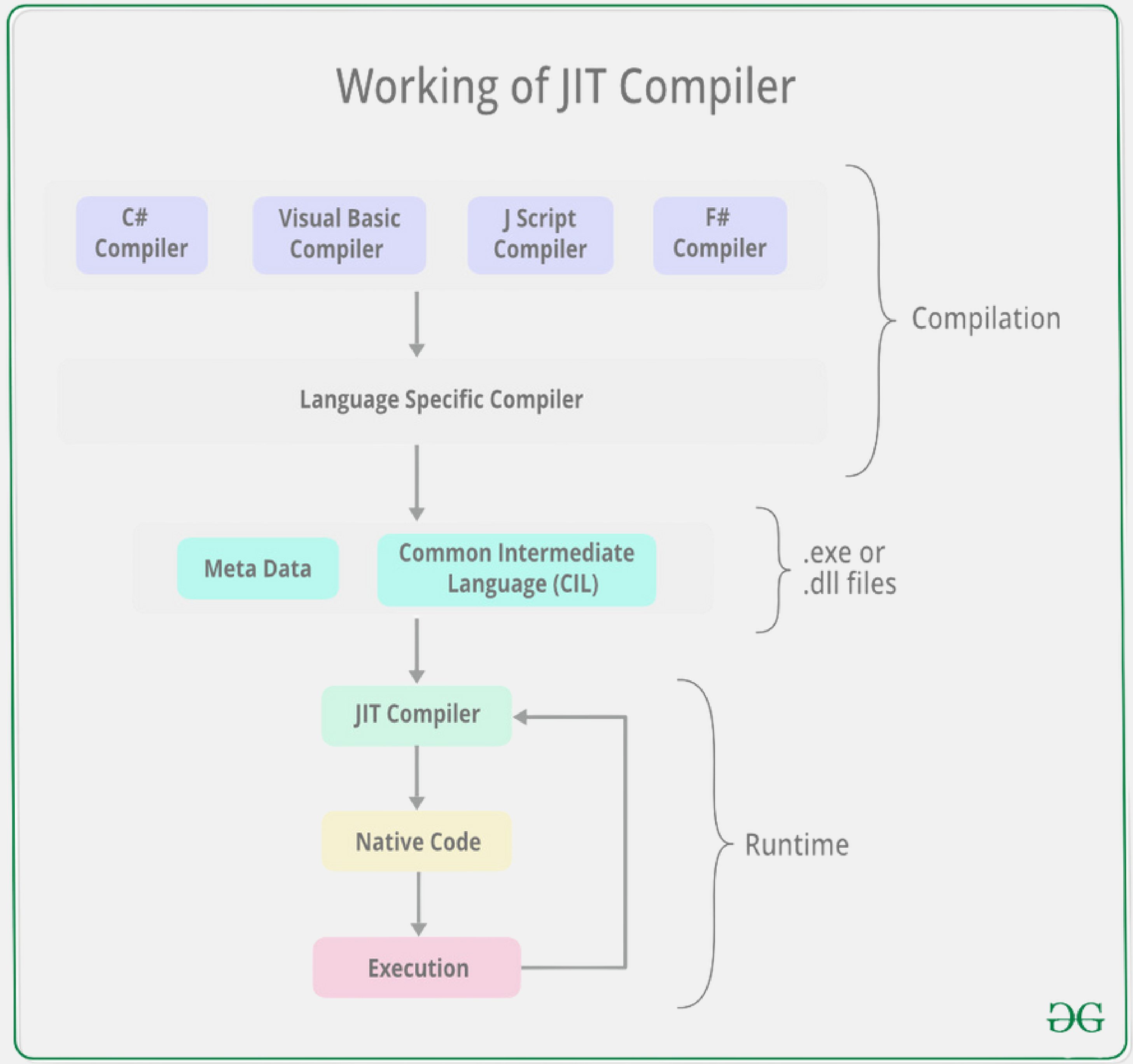
- N qubit system is represented by 2^N complex numbers
- Deploying a quantum computing algorithms means to apply unitary operators via matrix multiplication
- A good simulator corresponds to a good engine to perform linear algebra operations

Full state vector simulation in a nutshell

- N qubit system is represented by 2^N complex numbers
- Deploying a quantum computing algorithms means to apply unitary operators via matrix multiplication
- A good simulator corresponds to a good engine to perform linear algebra operations

What can we do to improve performances

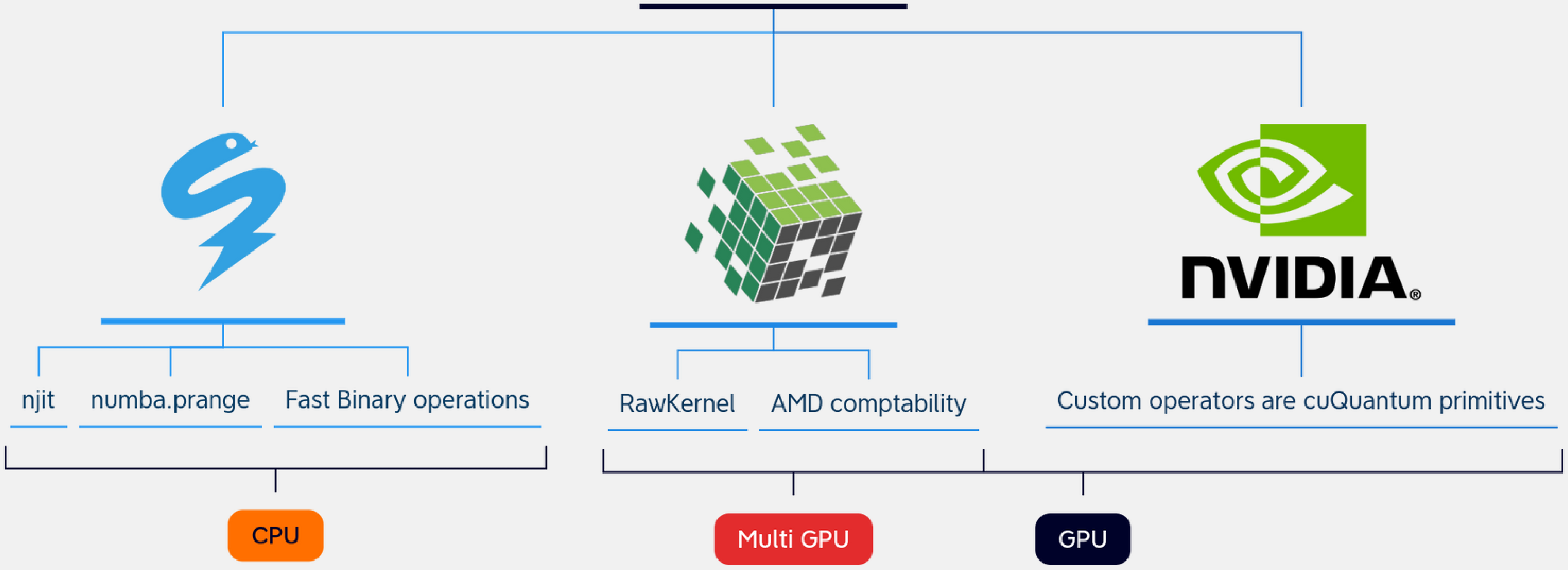
- Parallelize using multi threading
- Accelerate through GPUs
- Operation needs to be optimized



What is just-in-time compilation?

- A method for improving the performance of interpreted languages.
- We can exploit some frameworks in Python to achieve better performances compared to naive approaches

Qibojit



njit

numba.prange

Fast Binary operations

RawKernel

AMD comptability

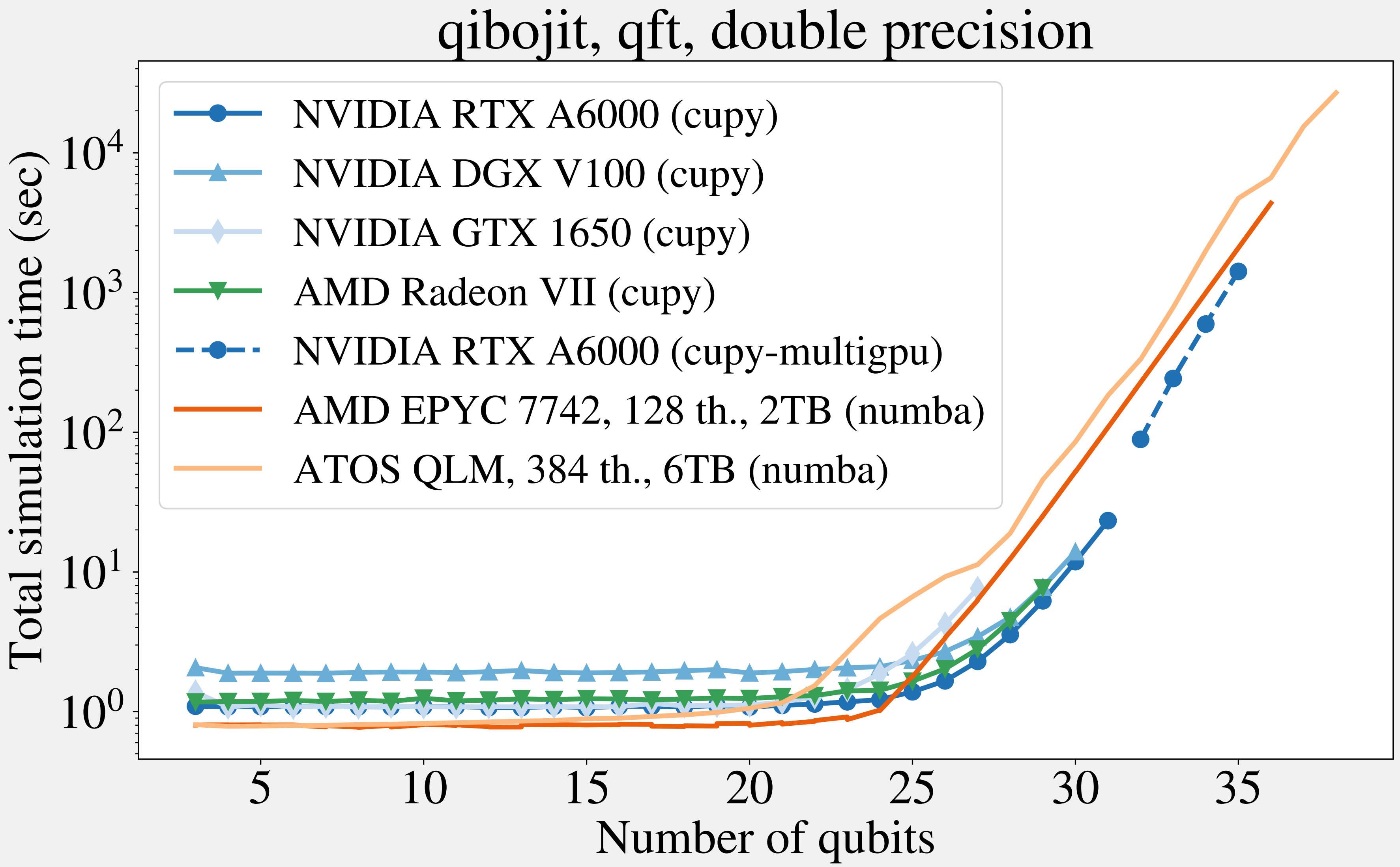
Custom operators are cuQuantum primitives

CPU

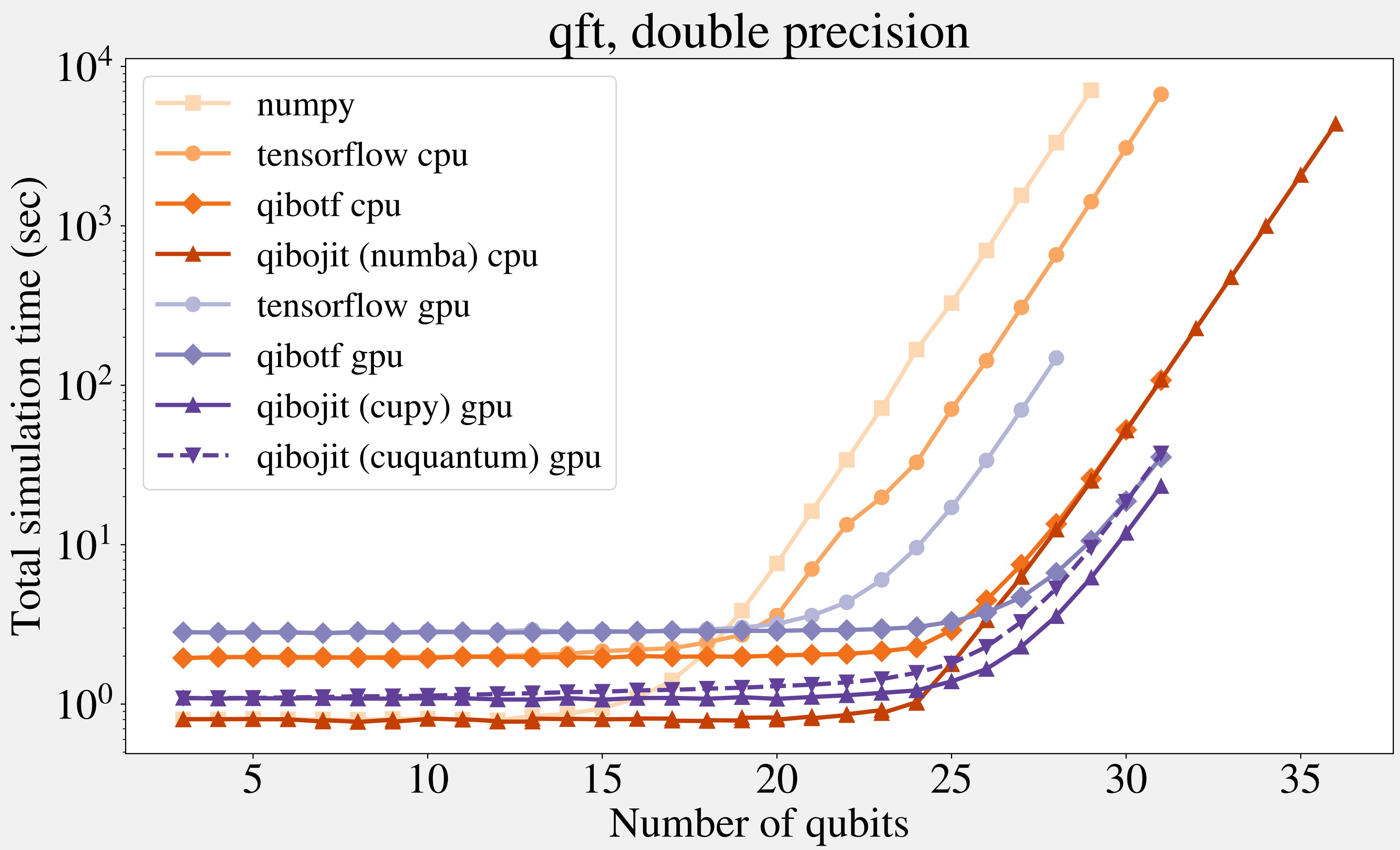
Multi GPU

GPU

Some benchmarks



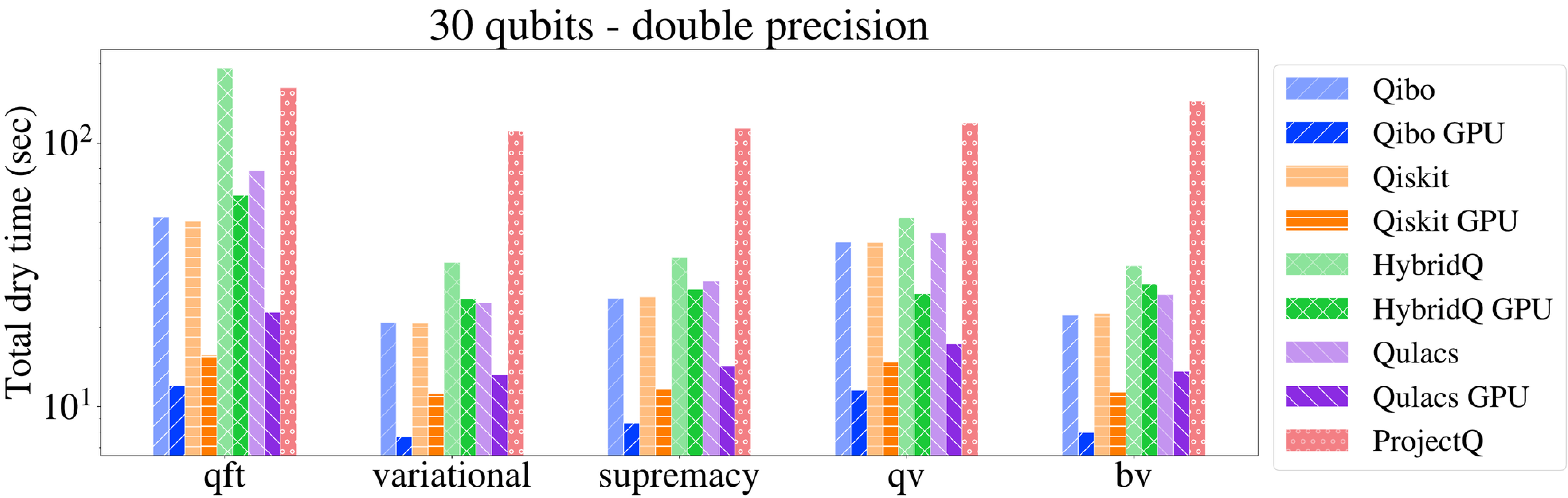
Total simulation time scaling with the number of qubits for simulating the qft circuit on different devices.



Total simulation time scaling with the number of qubits for simulating the qft circuit using different Qibo backends

https://github.com/qiboteam/qibojit-benchmarks

Comparison with other libraries



Total dry run time for simulating different circuits of 30 qubits, using libraries that support single (top) and double (bottom) precision.

TensorNetworks

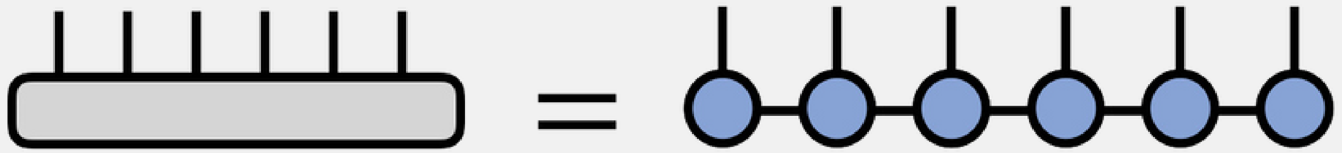
Approximate circuit execution using qibotn

https://github.com/qiboteam/qibotn

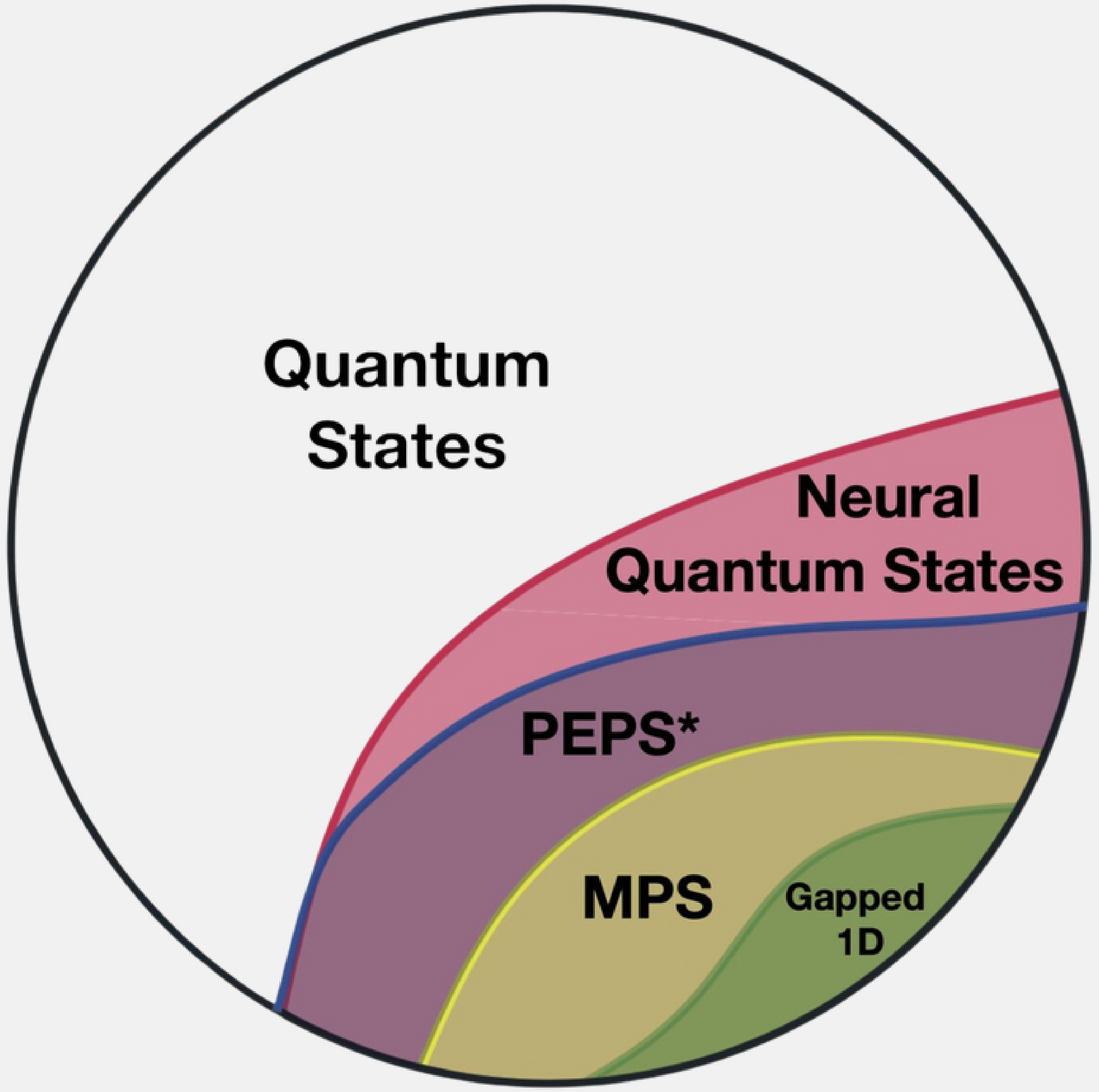
Tensor Networks

Tensor Networks are a powerful method for simulating quantum circuits by representing the state
or operator as a network of smaller tensors.

$$|\psi\rangle = \sum_{\sigma_1,\ldots,\sigma_L} A^{\sigma_1} A^{\sigma_2} \ldots A^{\sigma_{L-1}} A^{\sigma_L} |\sigma_1,\ \ldots,\sigma_L\rangle$$



It is an approximate method that enable to simulate circuits with polynomial complexity.



The expressive power of neural networks in quantum physics. Credits to G. Carleo.

U. Schollwoeck, Rev. Mod. Phys. 77, 259 (2005)

R. Orus, Annals of Physics 349 (2014) 117-158

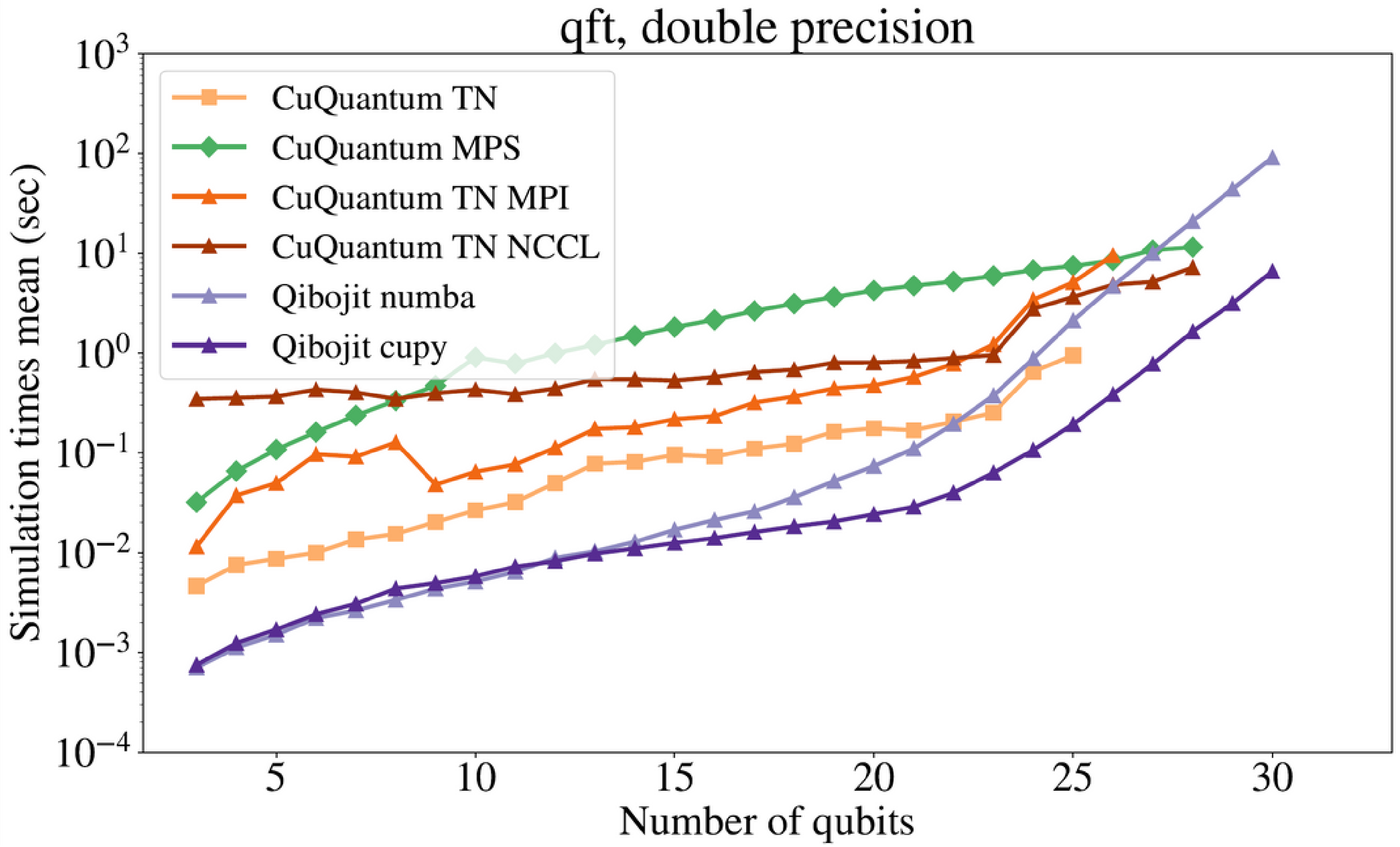
Qibotn

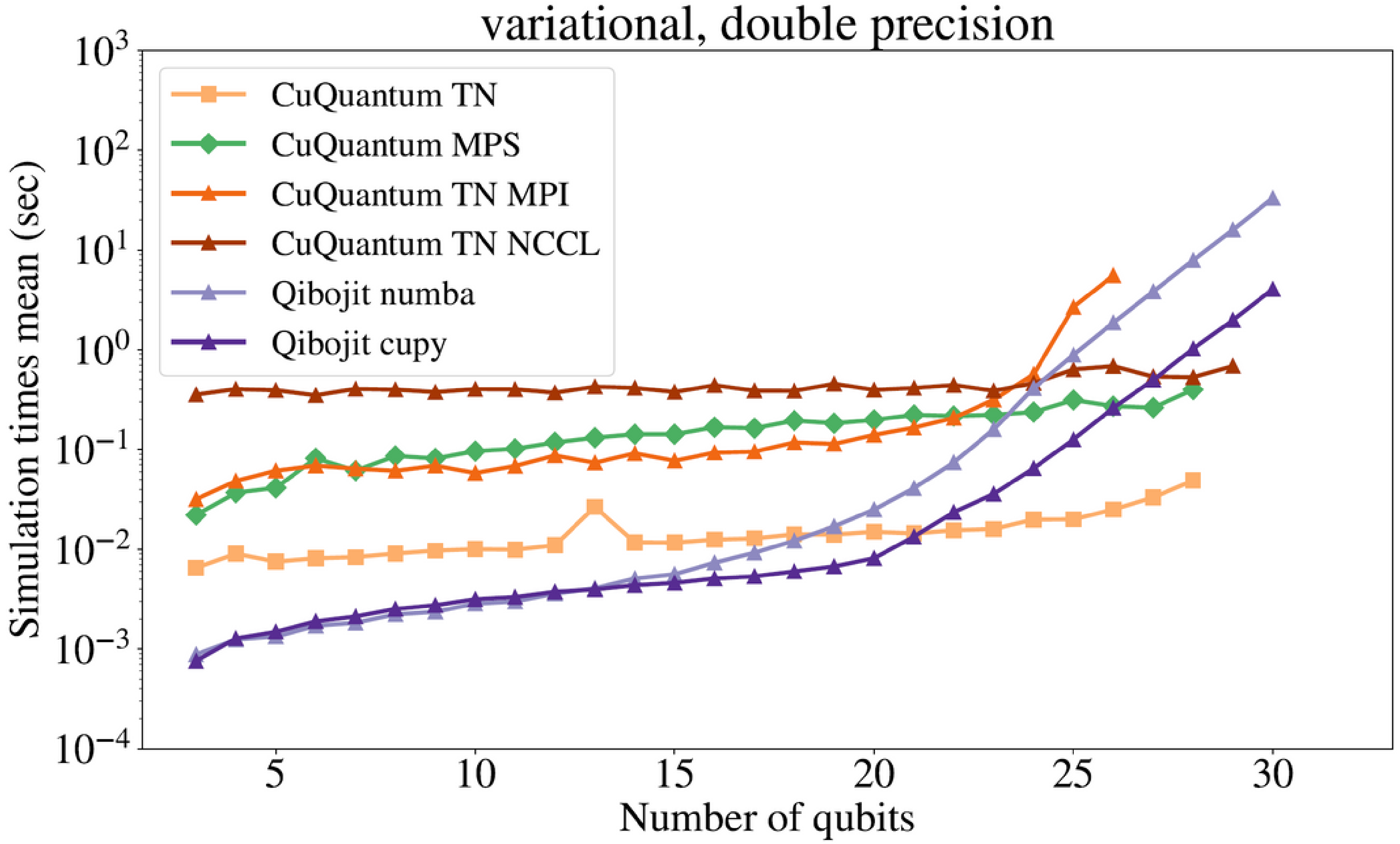
Supported Tensor Networks type:

- TensorNet (TN)
- Matrix Product States (MPS)

Supported HPC configurations:

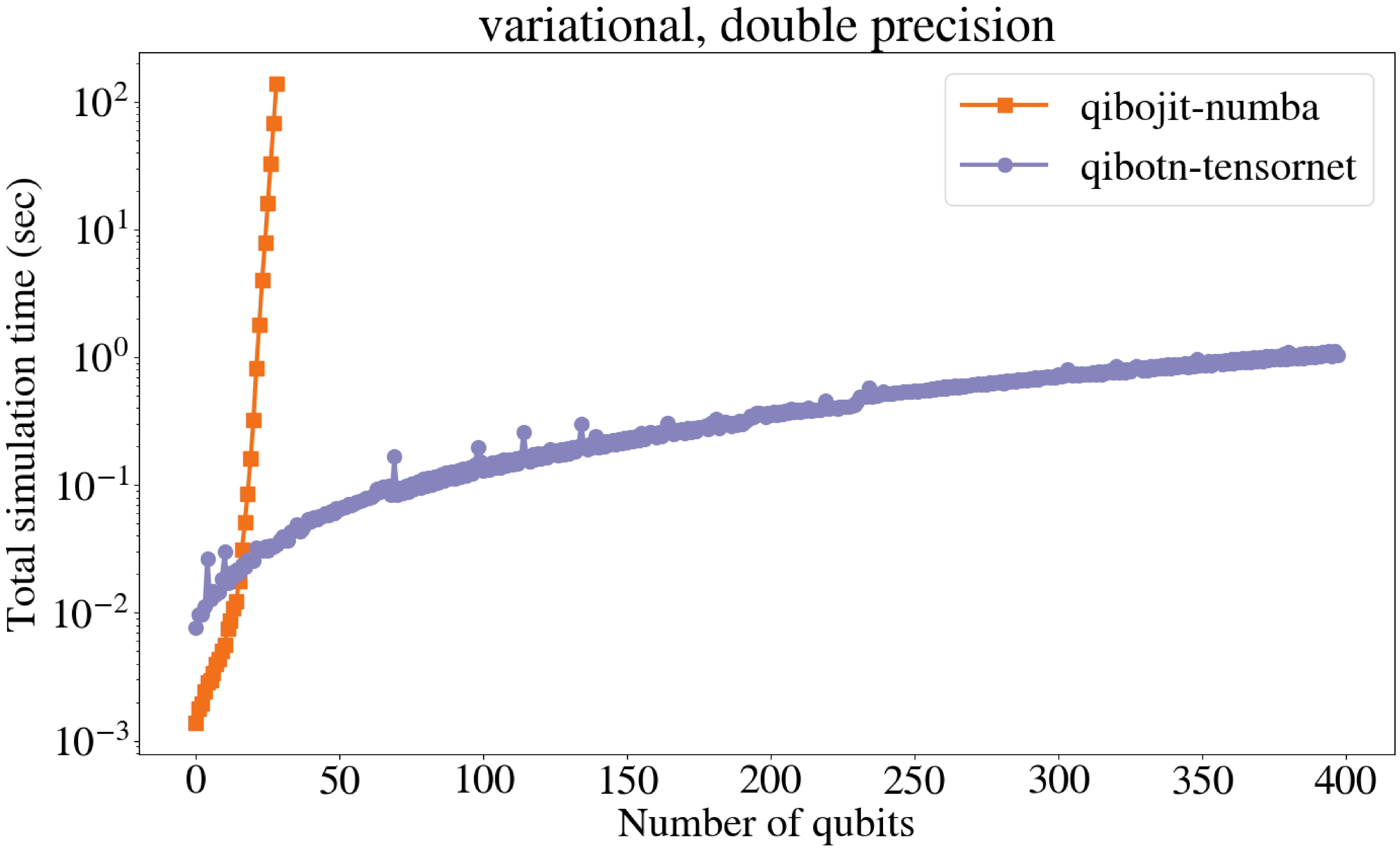
- single-node CPU
- single-node GPU or GPUs
- multi-node multi-GPU with Message Passing Interface (MPI)
- multi-node multi-GPU with NVIDIA Collective Communications Library (NCCL)





TN can scale up!

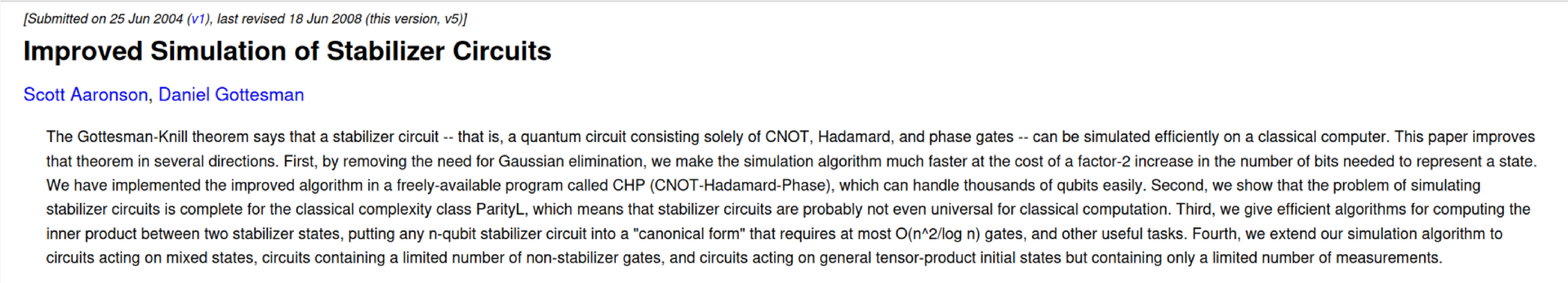
Running variational circuits with 400 qubits using A100 GPU



Clifford simulation

Speeding up execution for Clifford circuits

Clifford Simulation



[Submitted on 25 Jun 2004 (v1), last revised 18 Jun 2008 (this version, v5)]

Improved Simulation of Stabilizer Circuits

Scott Aaronson, Daniel Gottesman

The Gottesman-Knill theorem says that a stabilizer circuit -- that is, a quantum circuit consisting solely of CNOT, Hadamard, and phase gates -- can be simulated efficiently on a classical computer. This paper improves that theorem in several directions. First, by removing the need for Gaussian elimination, we make the simulation algorithm much faster at the cost of a factor-2 increase in the number of bits needed to represent a state. We have implemented the improved algorithm in a freely-available program called CHP (CNOT-Hadamard-Phase), which can handle thousands of qubits easily. Second, we show that the problem of simulating stabilizer circuits is complete for the classical complexity class ParityL, which means that stabilizer circuits are probably not even universal for classical computation. Third, we give efficient algorithms for computing the inner product between two stabilizer states, putting any n-qubit stabilizer circuit into a "canonical form" that requires at most O(n^2/log n) gates, and other useful tasks. Fourth, we extend our simulation algorithm to circuits acting on mixed states, circuits containing a limited number of non-stabilizer gates, and circuits acting on general tensor-product initial states but containing only a limited number of measurements.

Efficiently simulate gate application and measurements sampling in the stabilizers state representation.

$$\left(\begin{array}{ccc|ccc|c}
x_{11} & \cdots & x_{1n} & z_{11} & \cdots & z_{1n} & r_1 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
x_{n1} & \cdots & x_{nn} & z_{n1} & \cdots & z_{nn} & r_n \\
\hline
x_{(n+1)1} & \cdots & x_{(n+1)n} & z_{(n+1)1} & \cdots & z_{(n+1)n} & r_{n+1} \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
x_{(2n)1} & \cdots & x_{(2n)n} & z_{(2n)1} & \cdots & z_{(2n)n} & r_{2n}
\end{array}\right)$$

The state is represented by a tableau of **binary** variables.

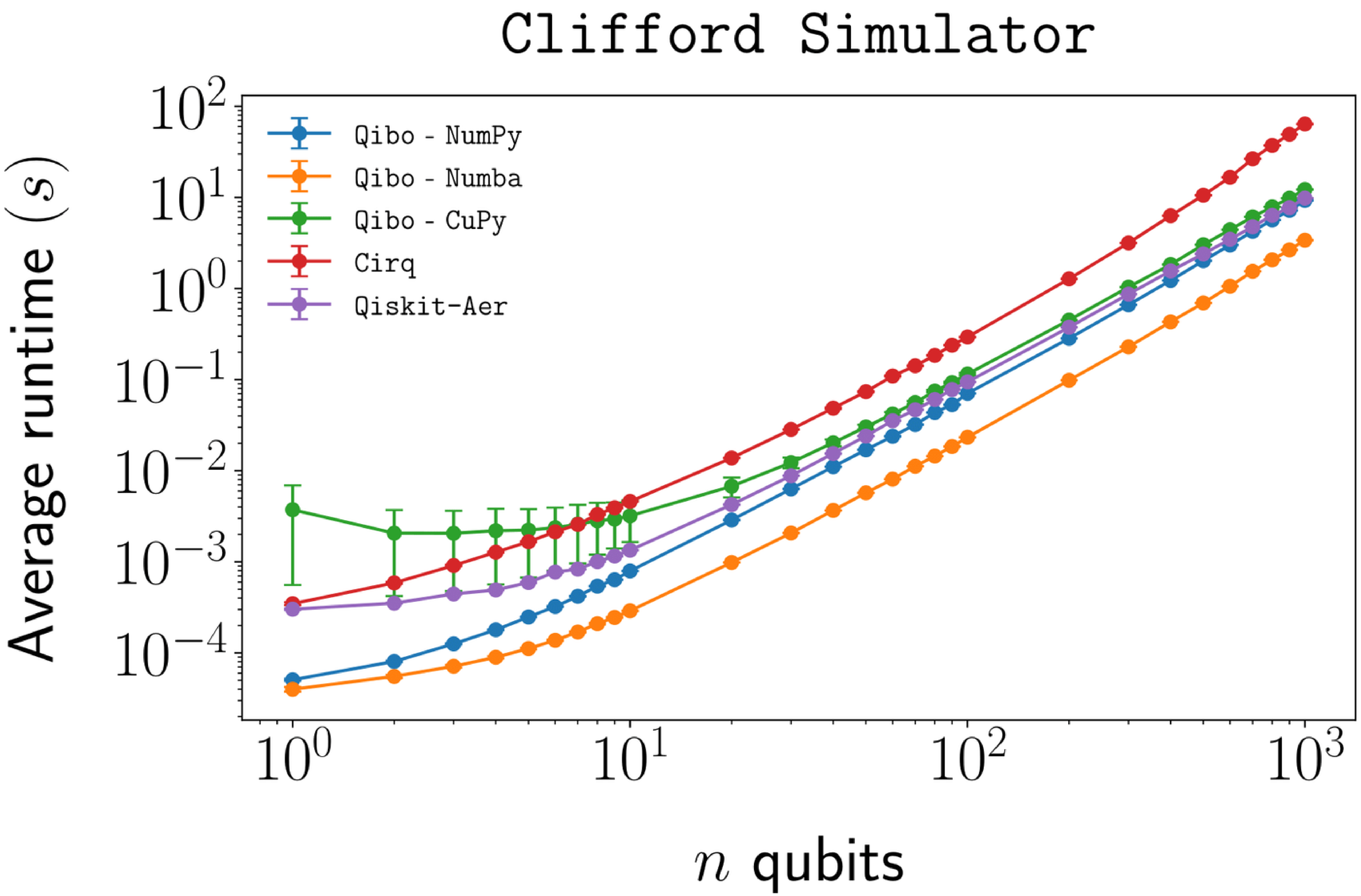
The complexity is:

- $\mathcal{O}(n)$ for unitary gates
- $\mathcal{O}(n^2)$ for measurements

Benchmarks

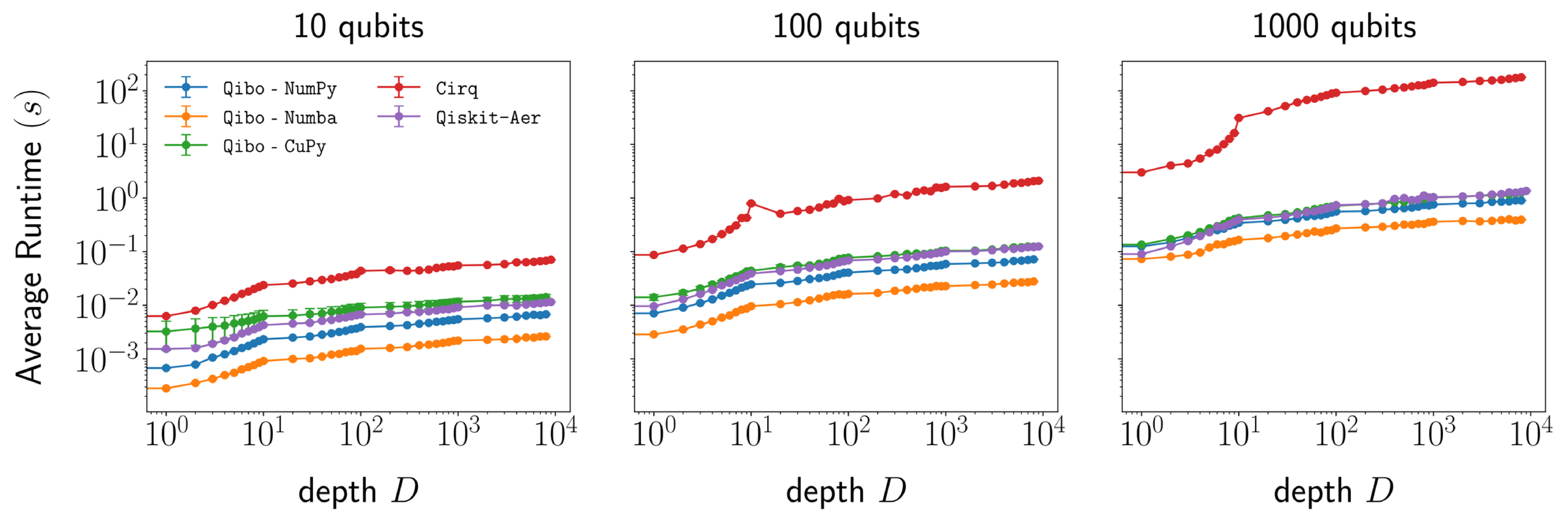
Papaluca et al, In preparation (2024)

Simulation of clifford circuits with an increasing number of qubits (no measurements). For each point we take the average over 100 different randomly generated circuits. Each circuit is generated following https://arxiv.org/abs/2003.0941, which guarantees an uniform distribution of the generated n-qubits clifford operators, but the depth is not fixed.



Benchmarks

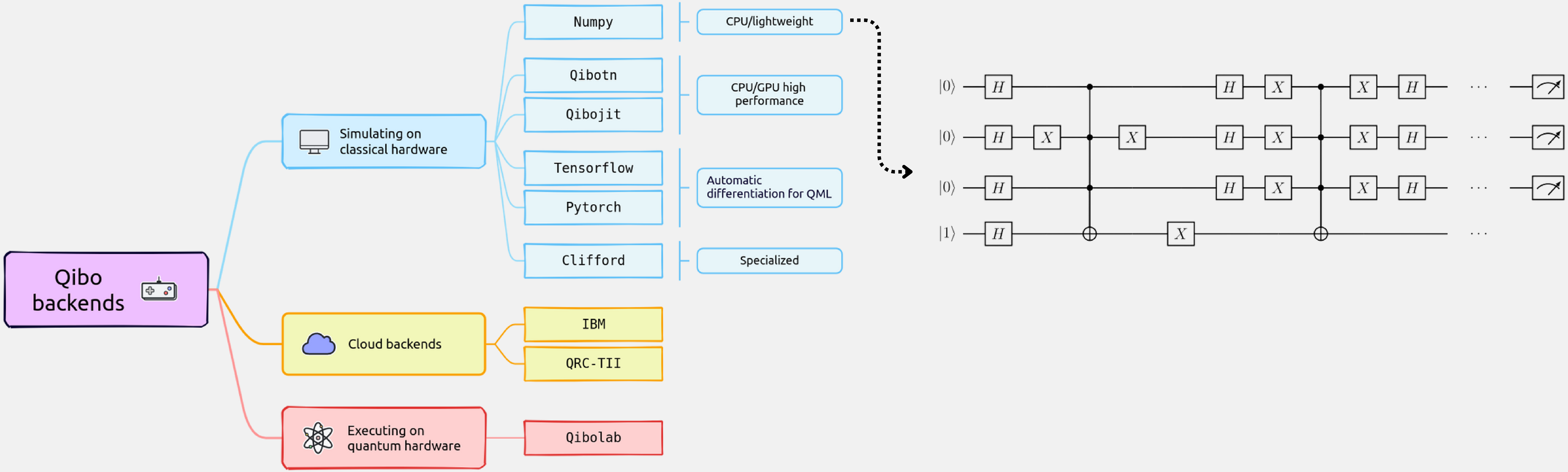
Papaluca et al, In preparation (2024)



Simulation of clifford circuits with an increasing depth for a fixed number of qubits (no measurements).For each point we take the average over 100 different randomly generated circuits. We randomly sample circuit moments composed by a single qubit cliffords layer followed by a two-qubits cliffords one. We sequentially stack up to 10 000 layers constructed this way.

Outlook

Quantum simulation summary



Qibo backends

Simulating on classical hardware

Numpy

Qibotn

Qibojit

Tensorflow

Pytorch

Clifford

CPU/lightweight

CPU/GPU high performance

Automatic differentiation for QML

Specialized

Cloud backends

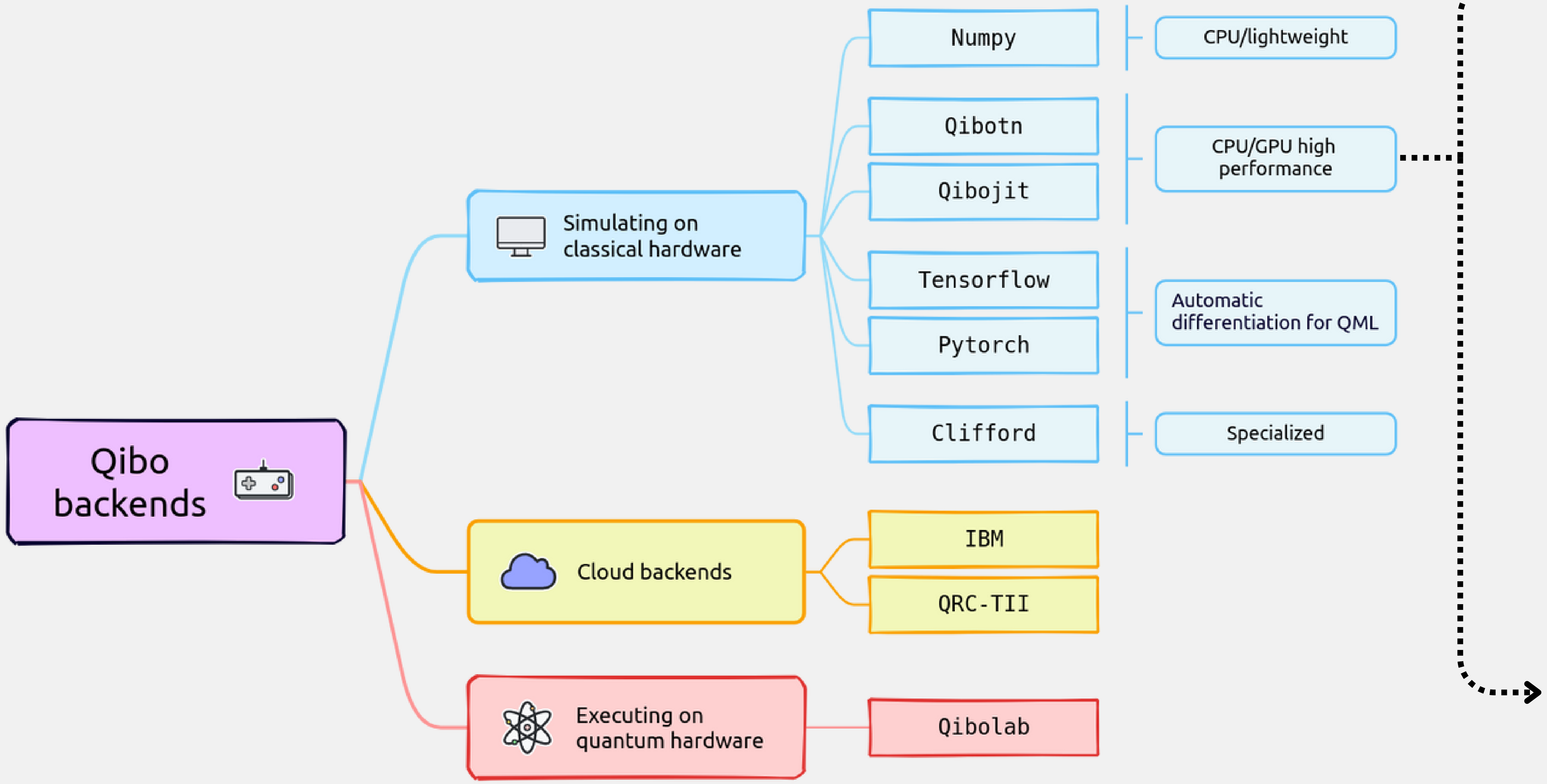
IBM

QRC-TII

Executing on quantum hardware

Qibolab

Quantum simulation summary



Qibo backends

Simulating on classical hardware

Numpy

CPU/lightweight

Qibotn

Qibojit

CPU/GPU high performance

Tensorflow

Pytorch

Automatic differentiation for QML

Clifford

Specialized

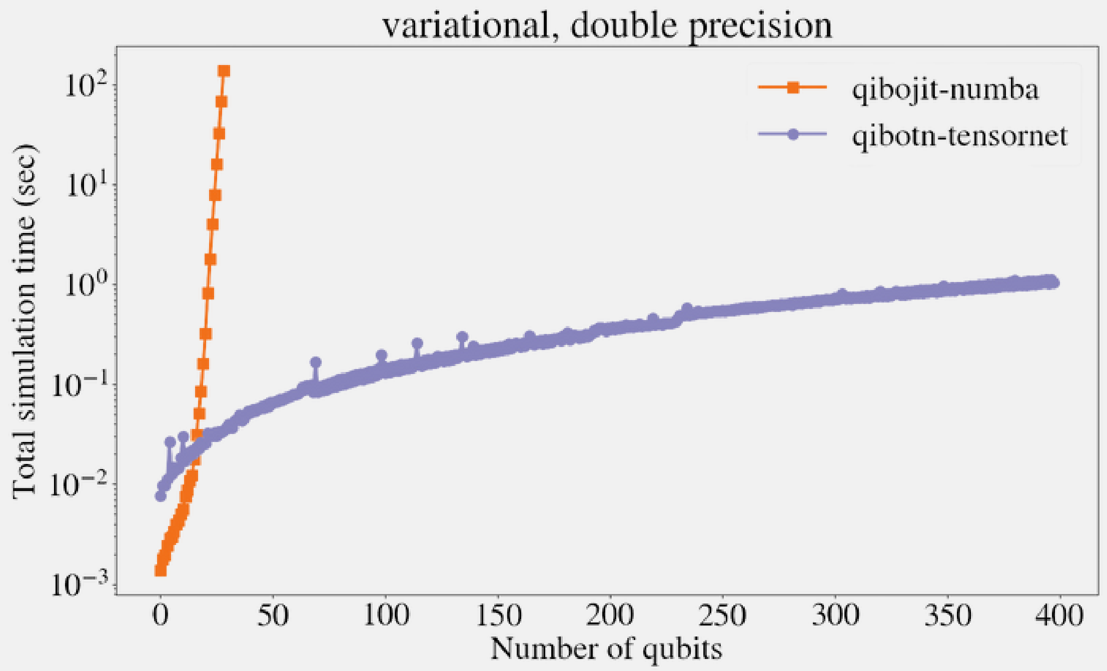
Cloud backends

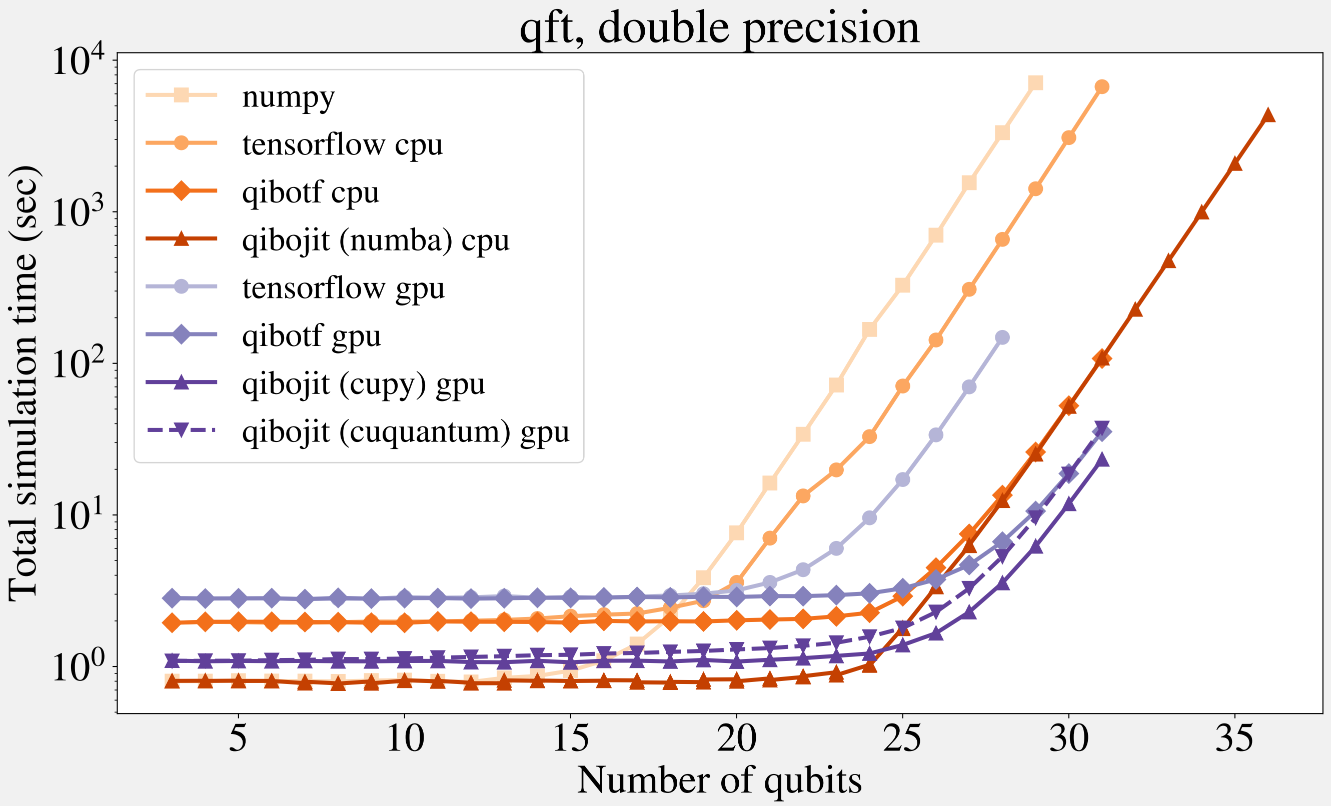
IBM

QRC-TII

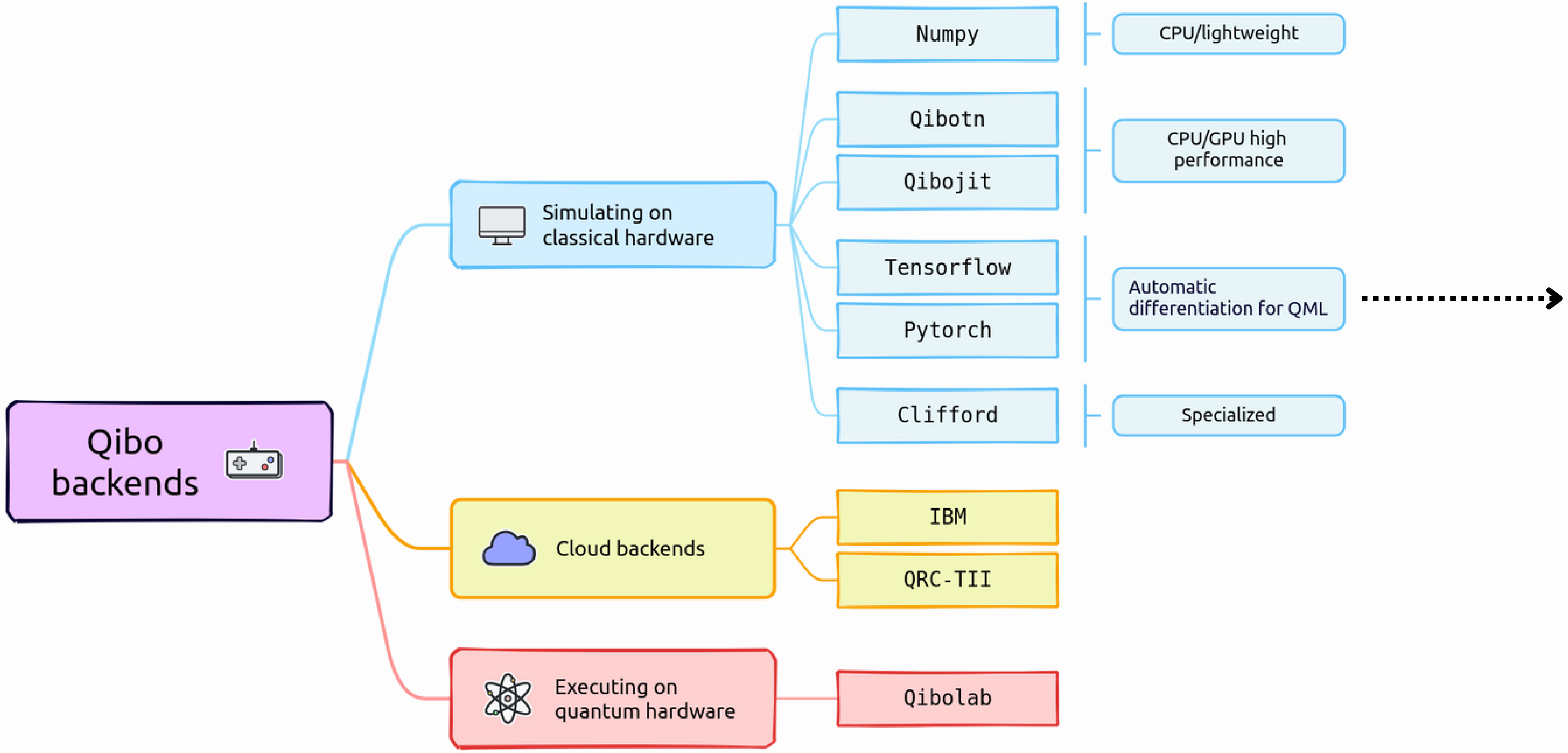
Executing on quantum hardware

Qibolab





Quantum simulation summary



Qibo backends

Simulating on classical hardware

Numpy

Qibotn

Qibojit

Tensorflow

Pytorch

Clifford

CPU/lightweight

CPU/GPU high performance

Automatic differentiation for QML

Specialized

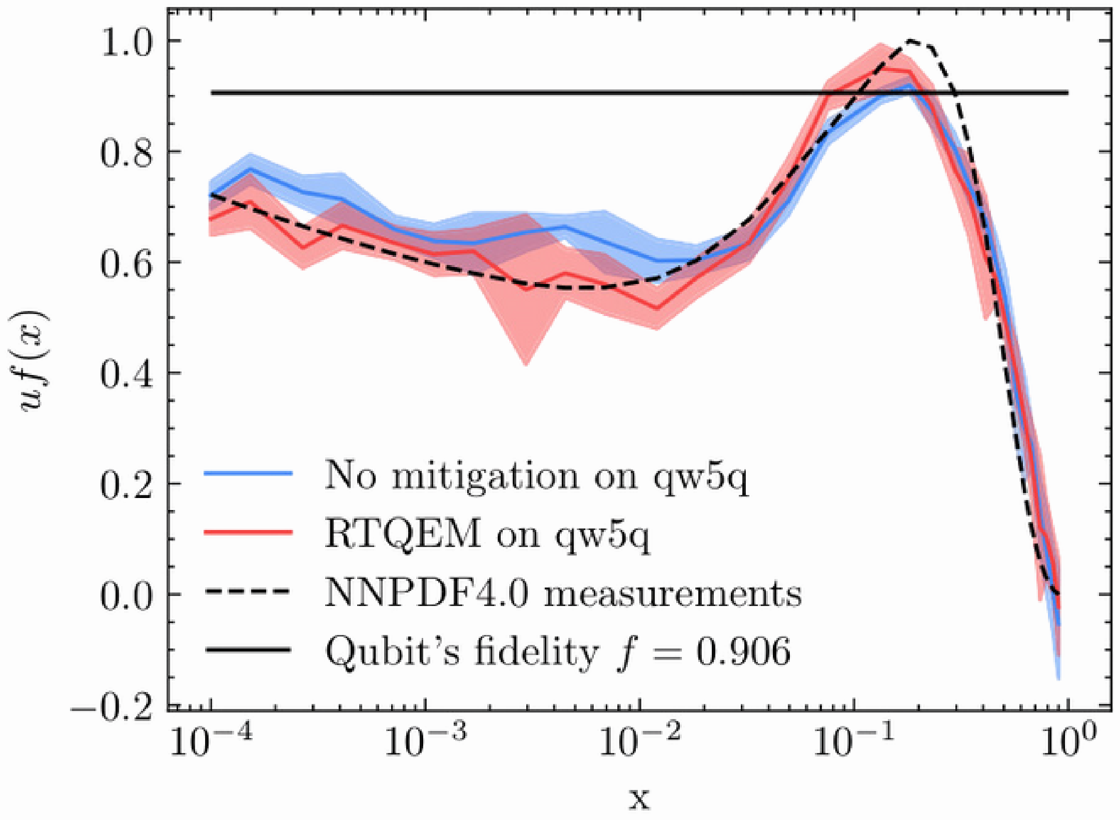
Cloud backends

IBM

QRC-TII

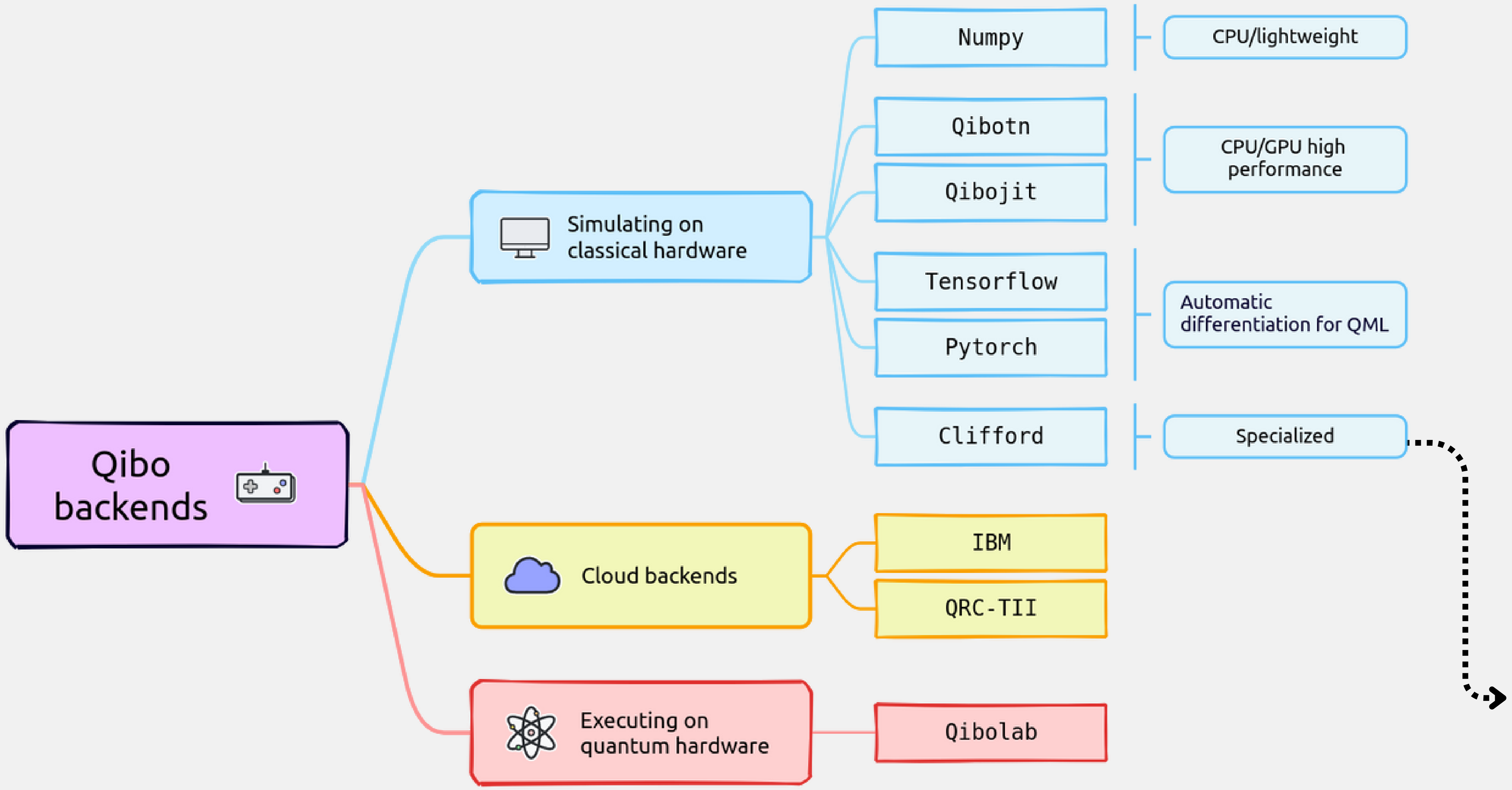
Executing on quantum hardware

Qibolab



arXiv:2311.05680

Quantum simulation summary



Qibo backends

Simulating on classical hardware

Numpy — CPU/lightweight

Qibotn, Qibojit — CPU/GPU high performance

Tensorflow, Pytorch — Automatic differentiation for QML

Clifford — Specialized

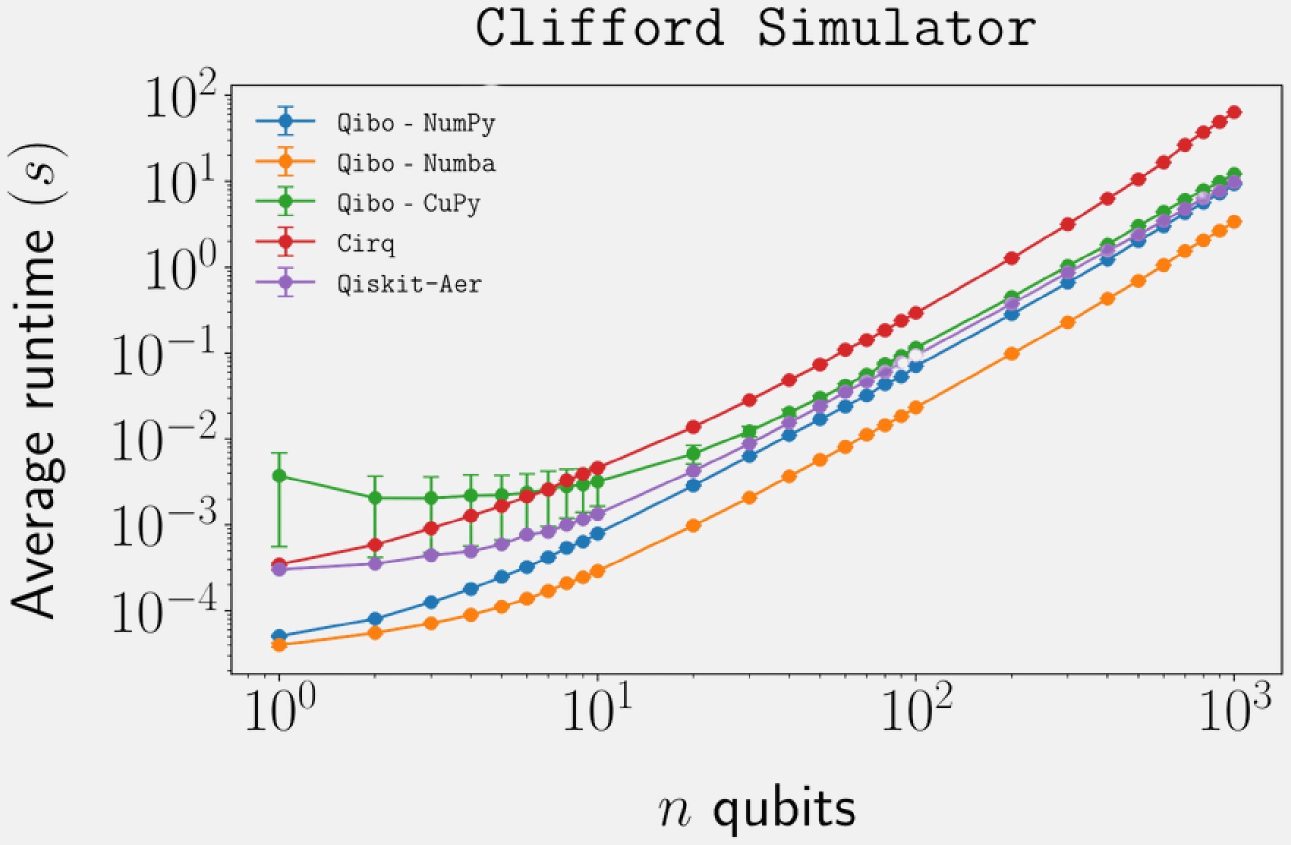
Cloud backends

IBM

QRC-TII

Executing on quantum hardware

Qibolab



Clifford Simulator

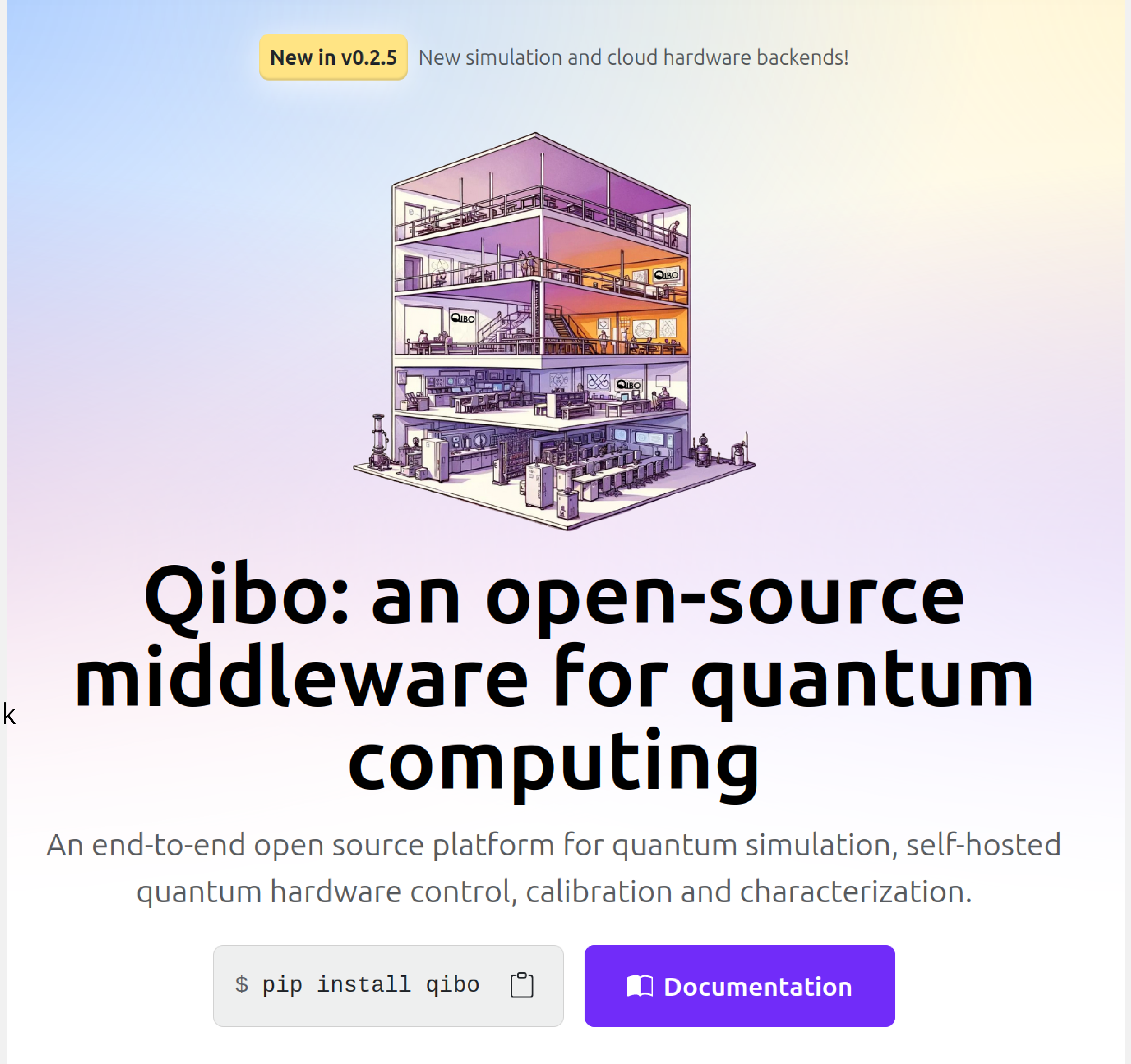
We have presented a **fully open source** quantum computing framework.

Qibo is compatible with state-of-the-art quantum simulation and offers several engines:

- **qibojit**: full-state vector simulation
- **qibotn**: tensor-network simulation
- **clifford**: specialized execution

But Qibo is **much more** than a quantum simulators:

- hardware execution and calibration -> Edoardo’s talk
- algorithms and error mitigation -> Matteo’s talk



New in v0.2.5 New simulation and cloud hardware backends!

Qibo: an open-source middleware for quantum computing

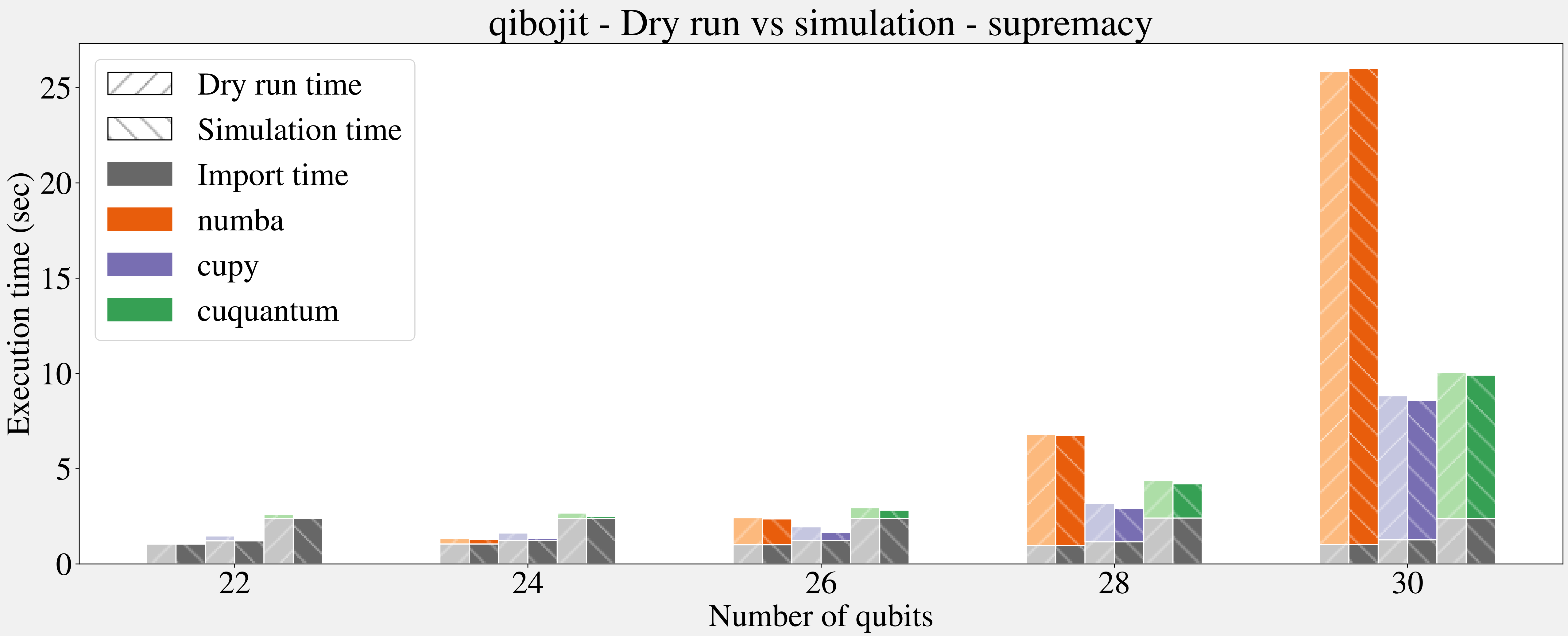
An end-to-end open source platform for quantum simulation, self-hosted quantum hardware control, calibration and characterization.

```
$ pip install qibo
```

Thanks for listening!

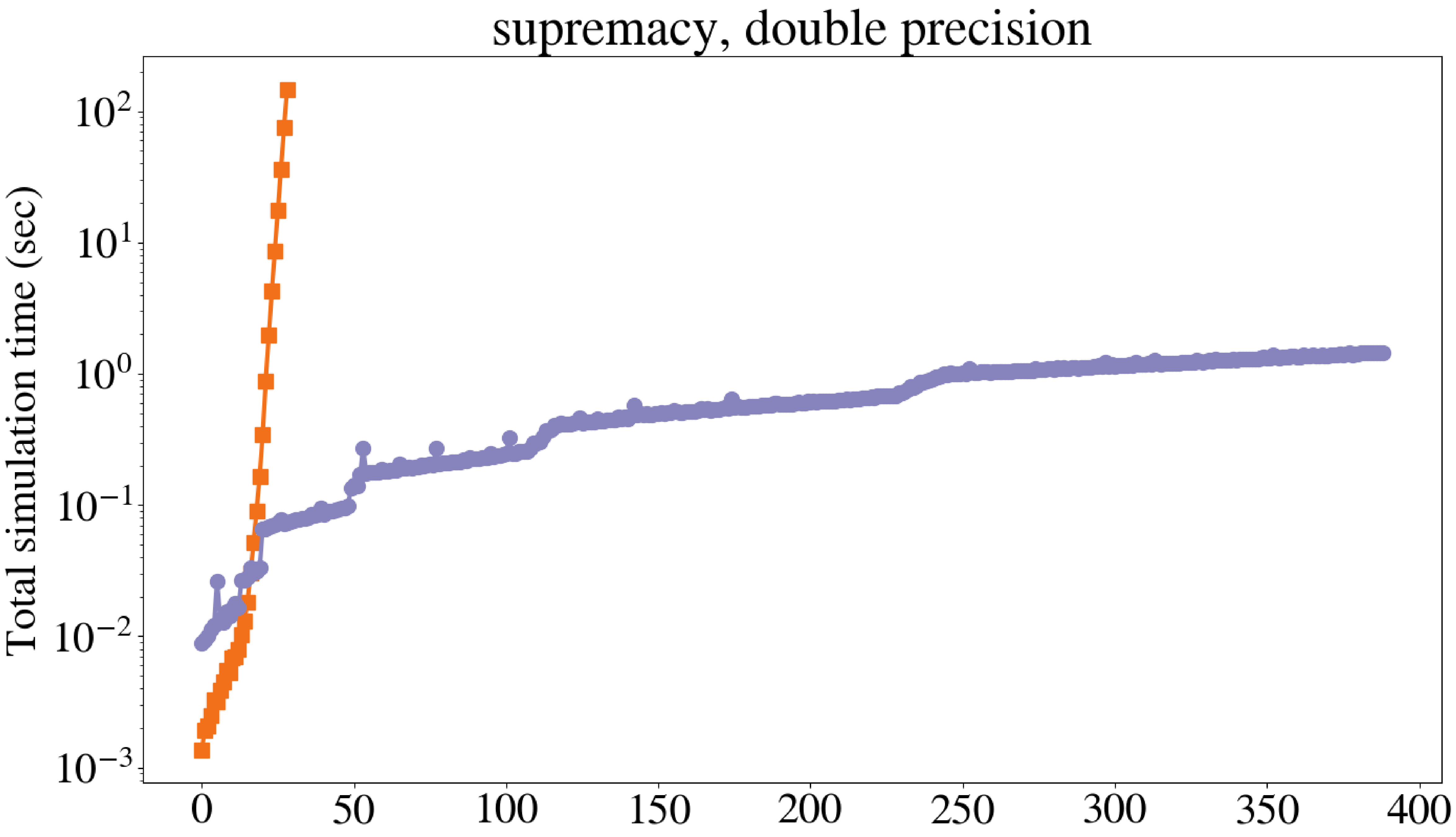
Questions?

Backup slides



Comparison between import, dry run and simulation times for the three platforms of the qibojit backend.

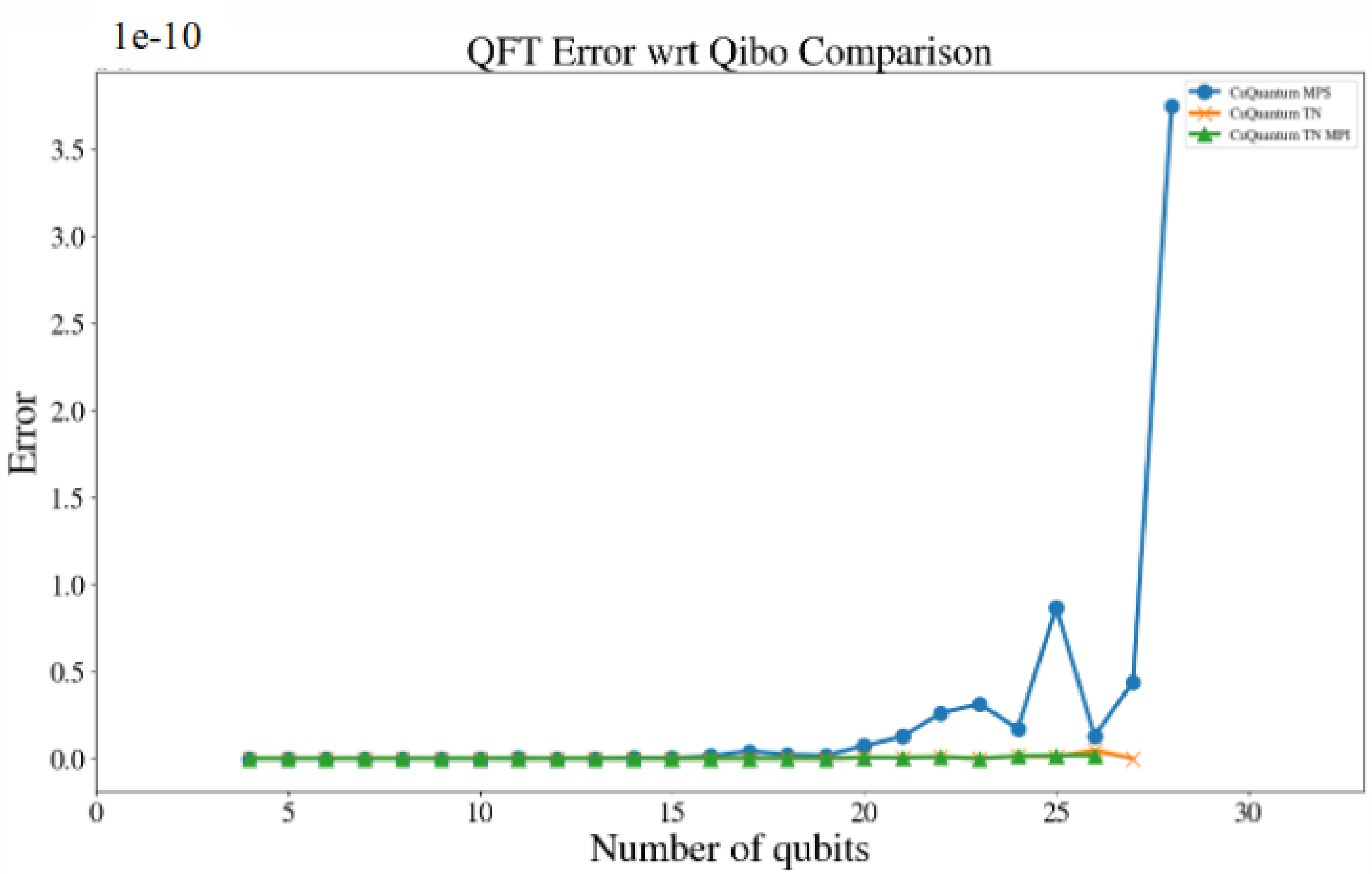
Running supremacy circuits with 400 qubits using A100 GPU



Accuracy comparison: tensor network vs state vector

Tensor Network accuracy is generally high.

MPS tends to have lower accuracy but acceptable.



$$\Delta = \sum_{i=0}^{2^N} |\psi_{\mathrm{Qibo}} - \psi_{\mathrm{Qibotn}}|$$