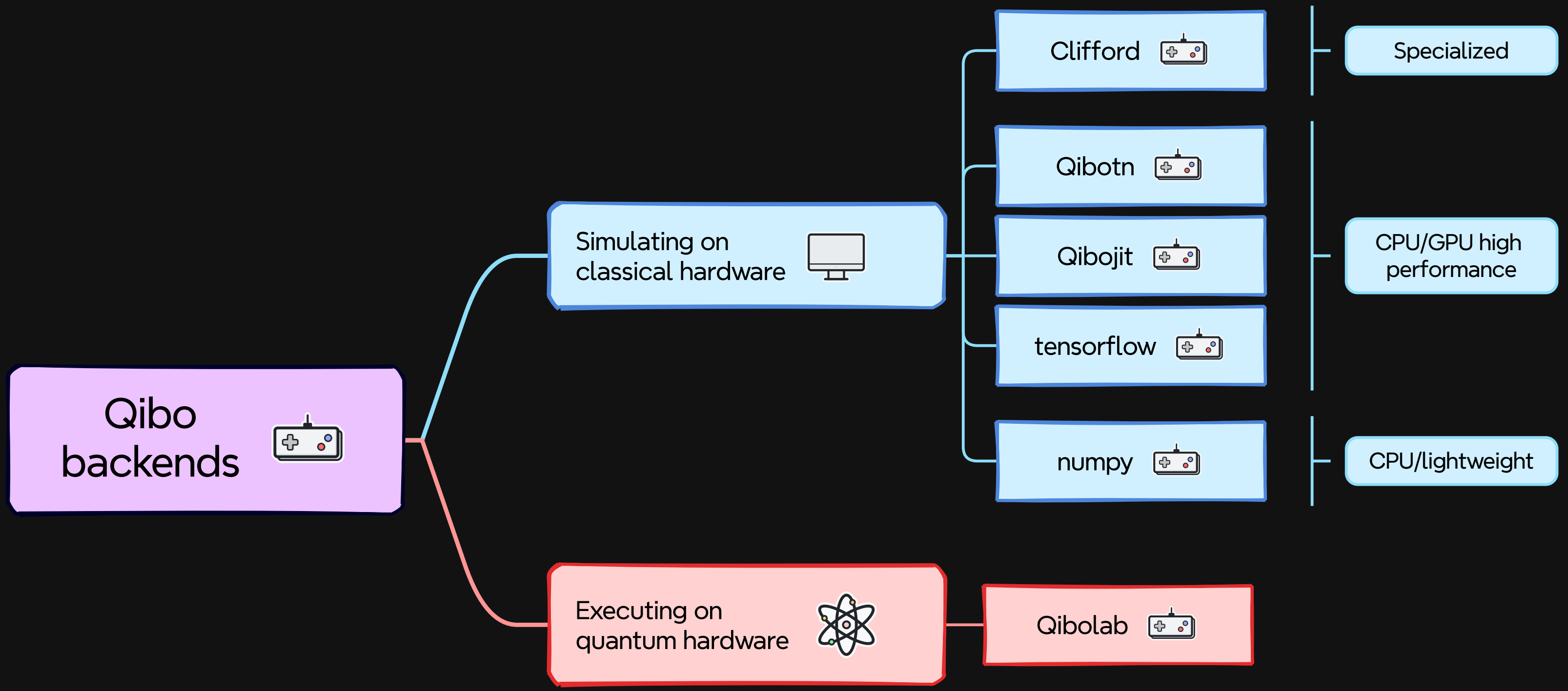
Simulation and Control

Quantum middleware

Alessandro Candido

Qibo [arXiv: 2009.01845]

Execution



State vector

Preserve whole information.

Challenges

- linear algebra
  - *i.e. array library*
- performances
- memory management

Approach

Adopt **widespread** and **optimized frameworks**, to benefit from their expertise (*software reuse*).

Chisel the last layer on top of each framework, to mold it on our use case.

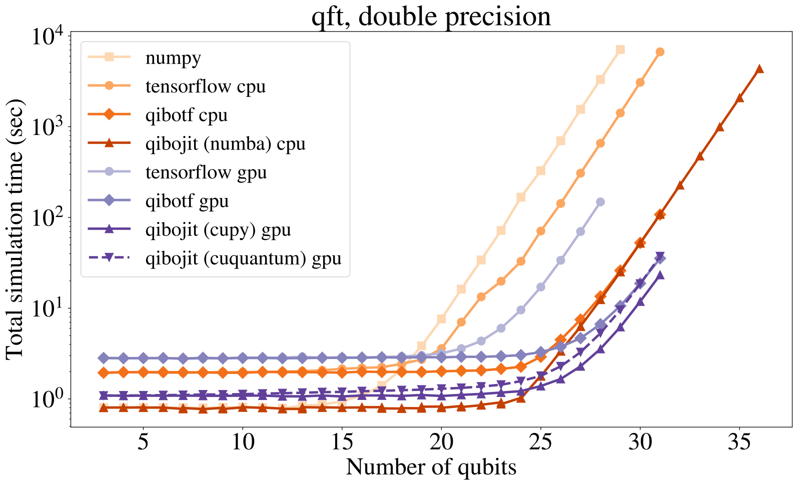
Backends mechanism

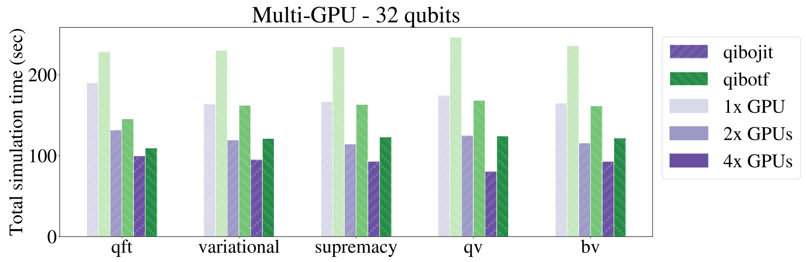
Plug the framework.

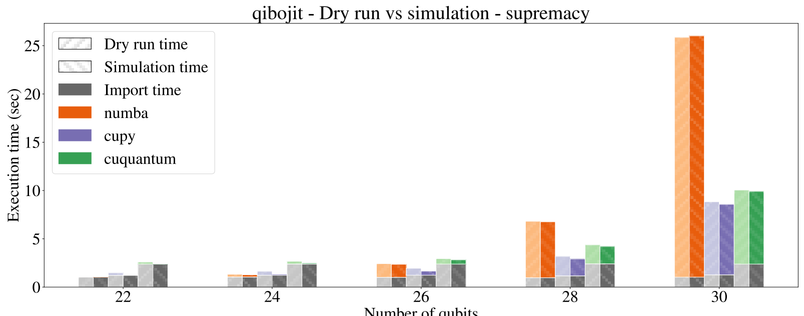
Structure the integration of the various libraries.

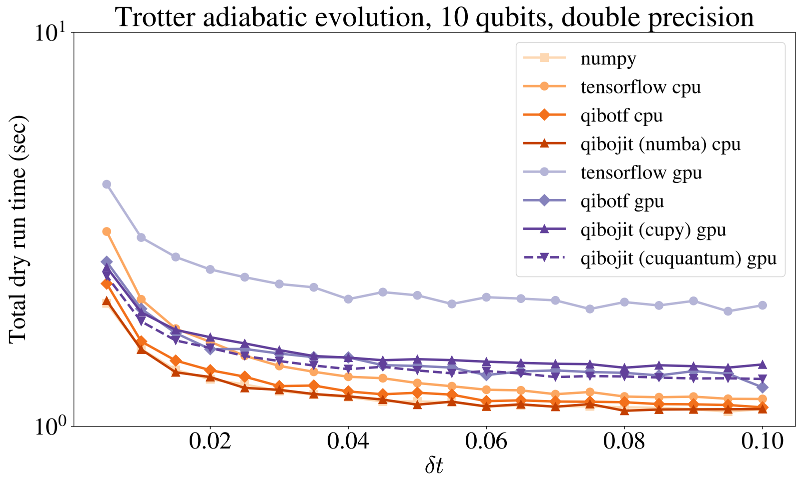
Common operations are implemented once and reused (when possible).

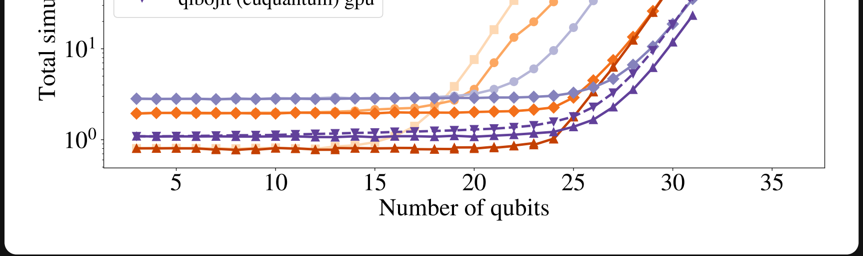
Results [arXiv: 2203.08826]

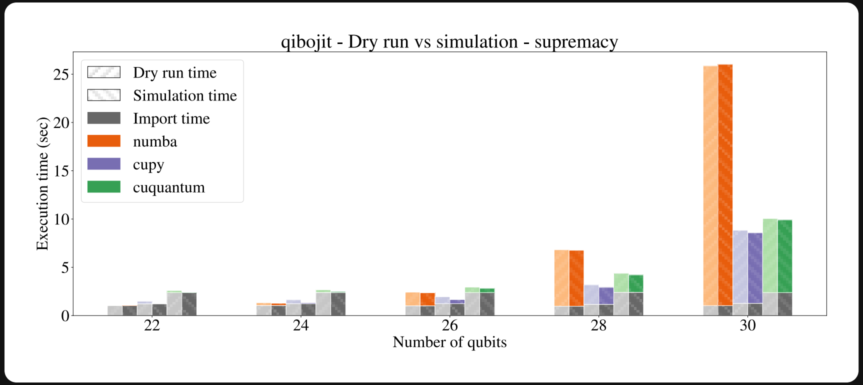


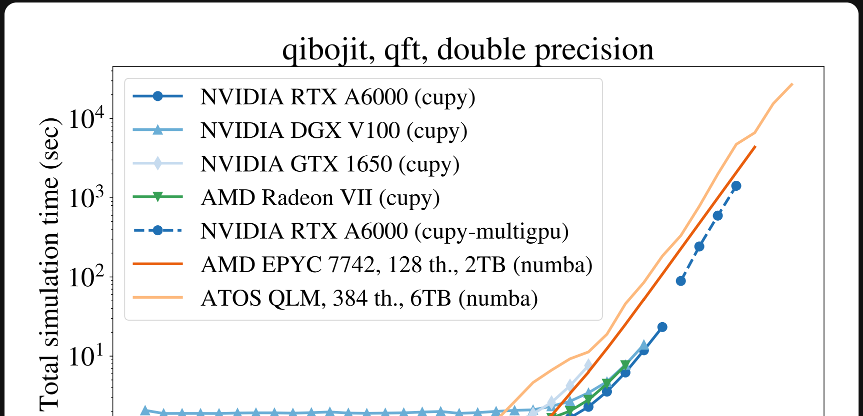


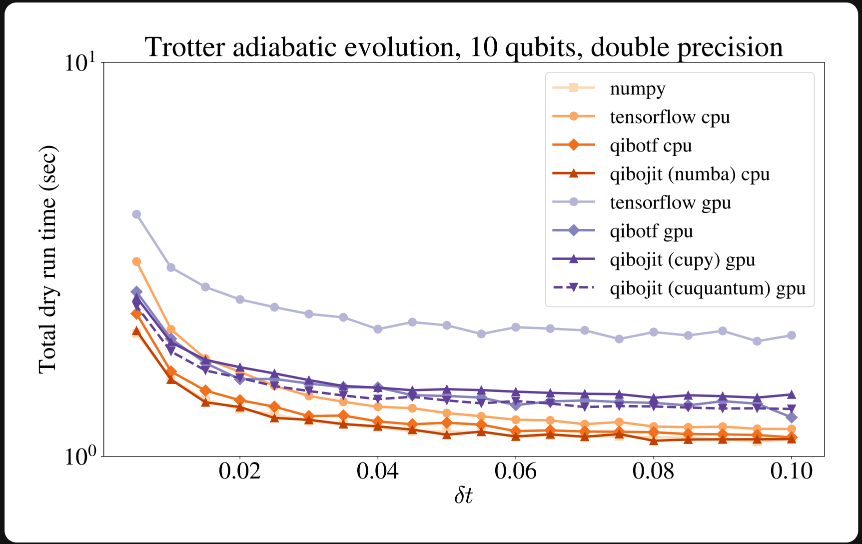


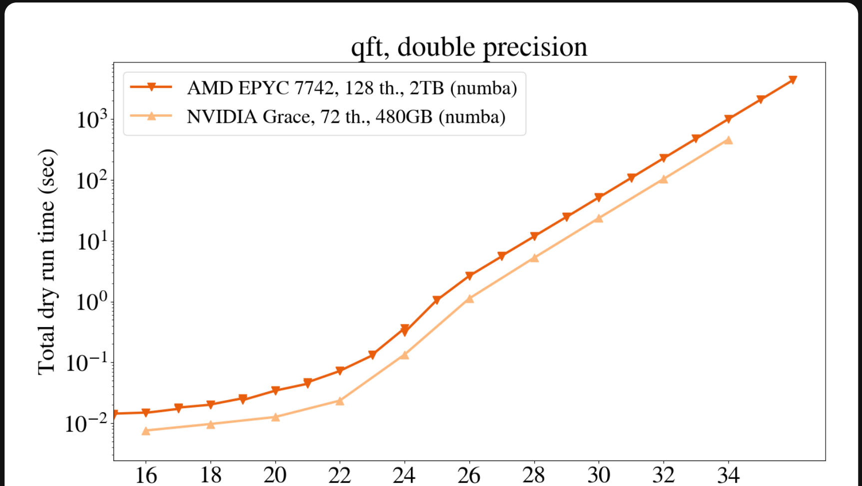


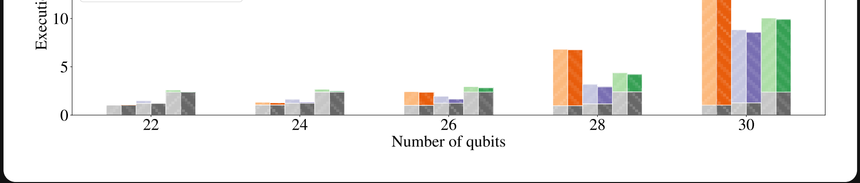

qibojit (cuquantum) gpu
Total simu
10^1
10^0
5
10
15
20
25
30
35
Number of qubits

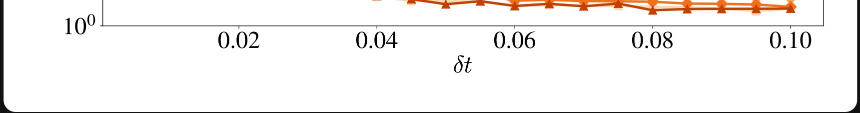

qibojit - Dry run vs simulation - supremacy
Dry run time
Simulation time
Import time
numba
cupy
cuquantum
Execution time (sec)
0
5
10
15
20
25
22
24
26
28
30
Number of qubits

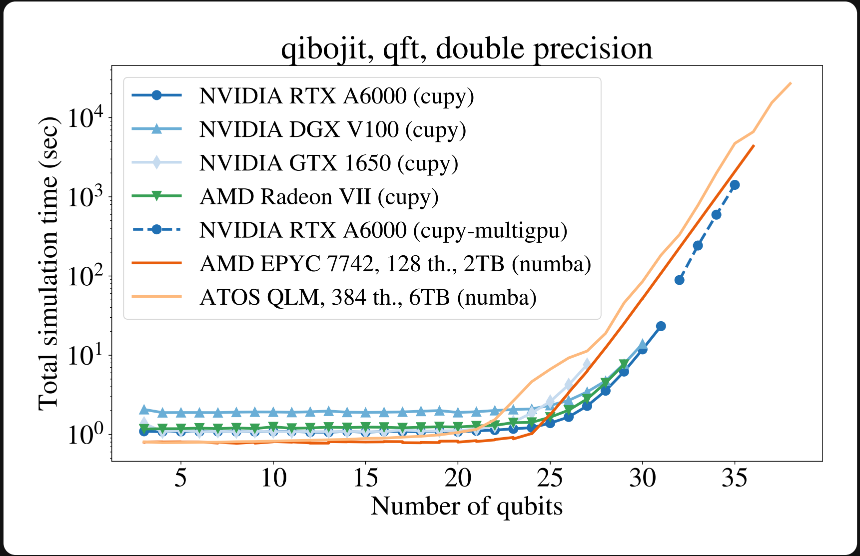

qibojit, qft, double precision
NVIDIA RTX A6000 (cupy)
NVIDIA DGX V100 (cupy)
NVIDIA GTX 1650 (cupy)
AMD Radeon VII (cupy)
NVIDIA RTX A6000 (cupy-multigpu)
AMD EPYC 7742, 128 th., 2TB (numba)
ATOS QLM, 384 th., 6TB (numba)
Total simulation time (sec)
10^4
10^3
10^2
10^1

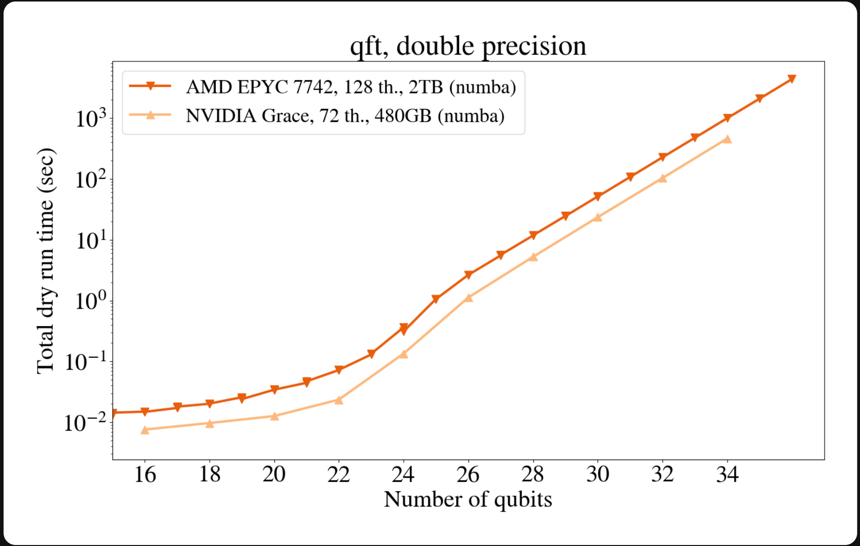

Trotter adiabatic evolution, 10 qubits, double precision
numpy
tensorflow cpu
qibotf cpu
qibojit (numba) cpu
tensorflow gpu
qibotf gpu
qibojit (cupy) gpu
qibojit (cuquantum) gpu
Total dry run time (sec)
10^1
10^0
0.02
0.04
0.06
0.08
0.10
δt

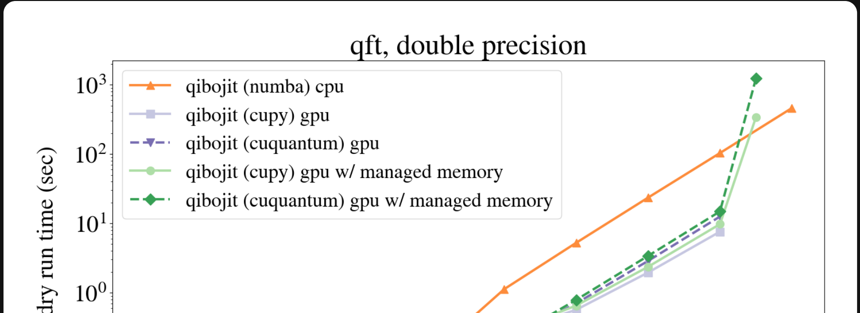

qft, double precision
AMD EPYC 7742, 128 th., 2TB (numba)
NVIDIA Grace, 72 th., 480GB (numba)
Total dry run time (sec)
10^3
10^2
10^1
10^0
10^{-1}
10^{-2}
16
18
20
22
24
26
28
30
32
34

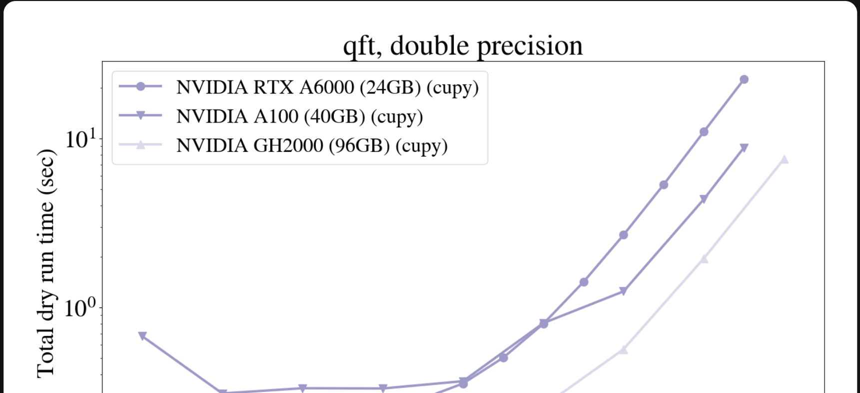

Executi
0
5
10
22
24
26
28
30
Number of qubits

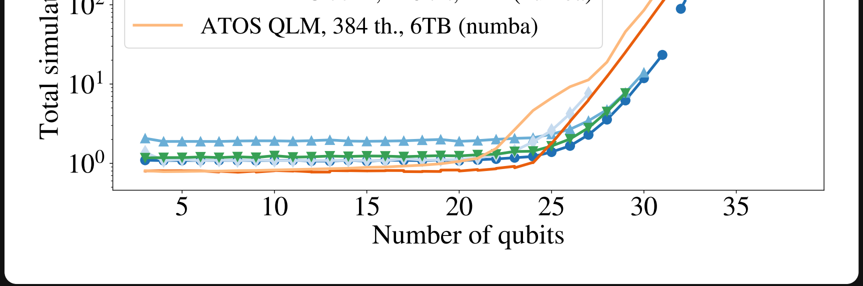

10^0
0.02
0.04
0.06
0.08
0.10
δt

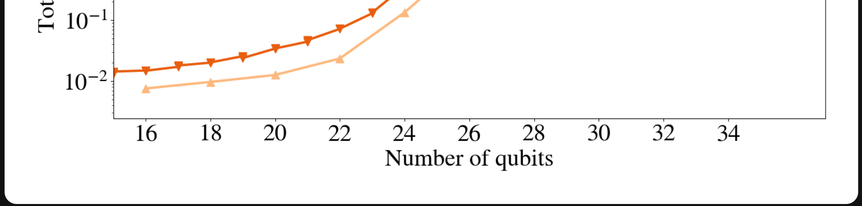

qibojit, qft, double precision
NVIDIA RTX A6000 (cupy)
NVIDIA DGX V100 (cupy)
NVIDIA GTX 1650 (cupy)
AMD Radeon VII (cupy)
NVIDIA RTX A6000 (cupy-multigpu)
AMD EPYC 7742, 128 th., 2TB (numba)
ATOS QLM, 384 th., 6TB (numba)
Total simulation time (sec)
Number of qubits

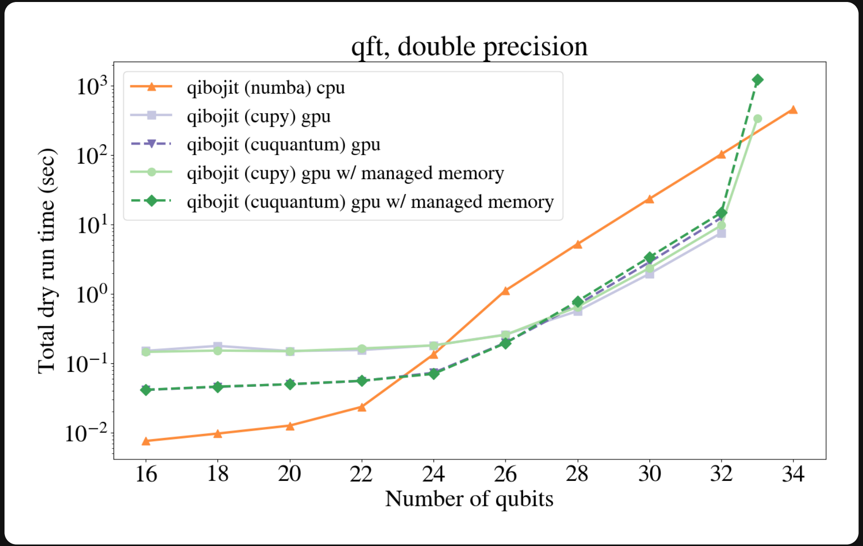

qft, double precision
AMD EPYC 7742, 128 th., 2TB (numba)
NVIDIA Grace, 72 th., 480GB (numba)
Total dry run time (sec)
Number of qubits

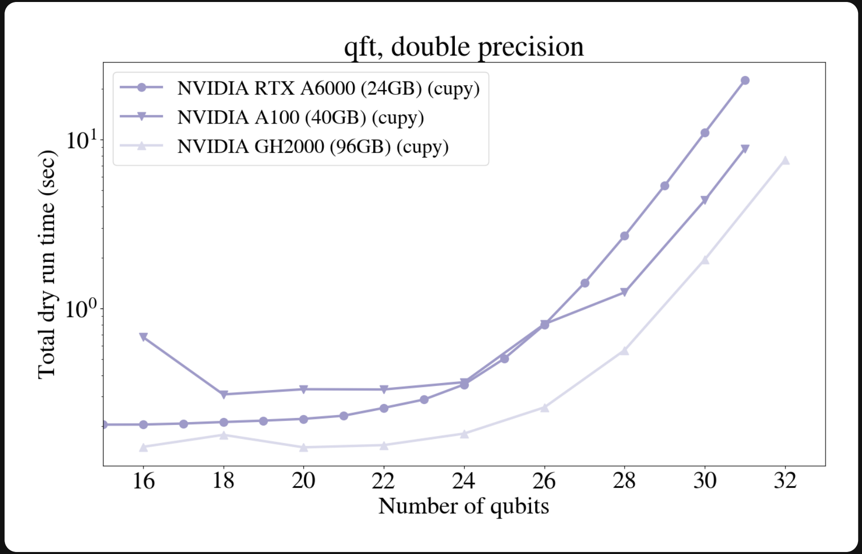

qft, double precision
qibojit (numba) cpu
qibojit (cupy) gpu
qibojit (cuquantum) gpu
qibojit (cupy) gpu w/ managed memory
qibojit (cuquantum) gpu w/ managed memory
dry run time (sec)


qft, double precision
NVIDIA RTX A6000 (24GB) (cupy)
NVIDIA A100 (40GB) (cupy)
NVIDIA GH2000 (96GB) (cupy)
Total dry run time (sec)


ATOS QLM, 384 th., 6TB (numba)
Total simula
Number of qubits


Number of qubits


qft, double precision
qibojit (numba) cpu
qibojit (cupy) gpu
qibojit (cuquantum) gpu
qibojit (cupy) gpu w/ managed memory
qibojit (cuquantum) gpu w/ managed memory
Total dry run time (sec)
Number of qubits


qft, double precision
NVIDIA RTX A6000 (24GB) (cupy)
NVIDIA A100 (40GB) (cupy)
NVIDIA GH2000 (96GB) (cupy)
Total dry run time (sec)
Number of qubits

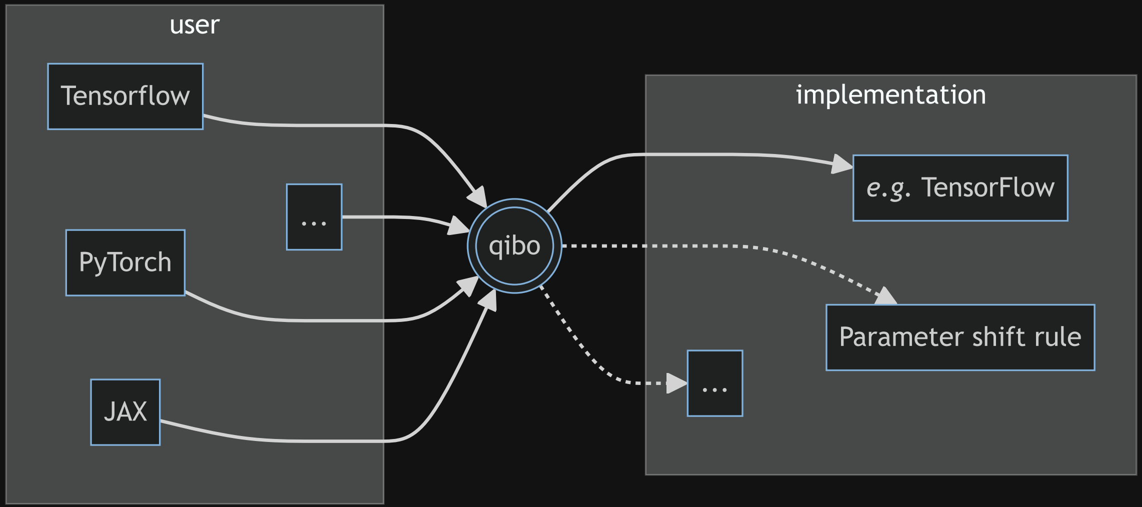
Automatic differentiation

for quantum machine learning (QML)

Autodiff simulation is fundamental to support QML investigation.

A dedicated differentiable backend in simulation can considerably help algorithms development.

Moving towards a single interface, encompassing both simulation and quantum hardware implementations.



Framework portability: implement in one, export derivatives.

Clifford

Specialized execution.

$$|\psi\rangle = U|\psi\rangle$$

Theorem 1 *Given an n-qubit state $|\psi\rangle$, the following are equivalent:*

(i) $|\psi\rangle$ can be obtained from $|0\rangle \otimes n$ by CNOT, Hadamard, and phase gates only.
(ii) $|\psi\rangle$ can be obtained from $|0\rangle \otimes n$ by CNOT, Hadamard, phase, and measurement gates only.
(iii) $|\psi\rangle$ is stabilized by exactly 2n Pauli operators.
*(iv) $|\psi\rangle$ **is uniquely determined by** $S(|\psi\rangle) = Stab(|\psi\rangle) \cap P_n$ **or the group of Pauli operators that stabilize** $|\psi\rangle$*

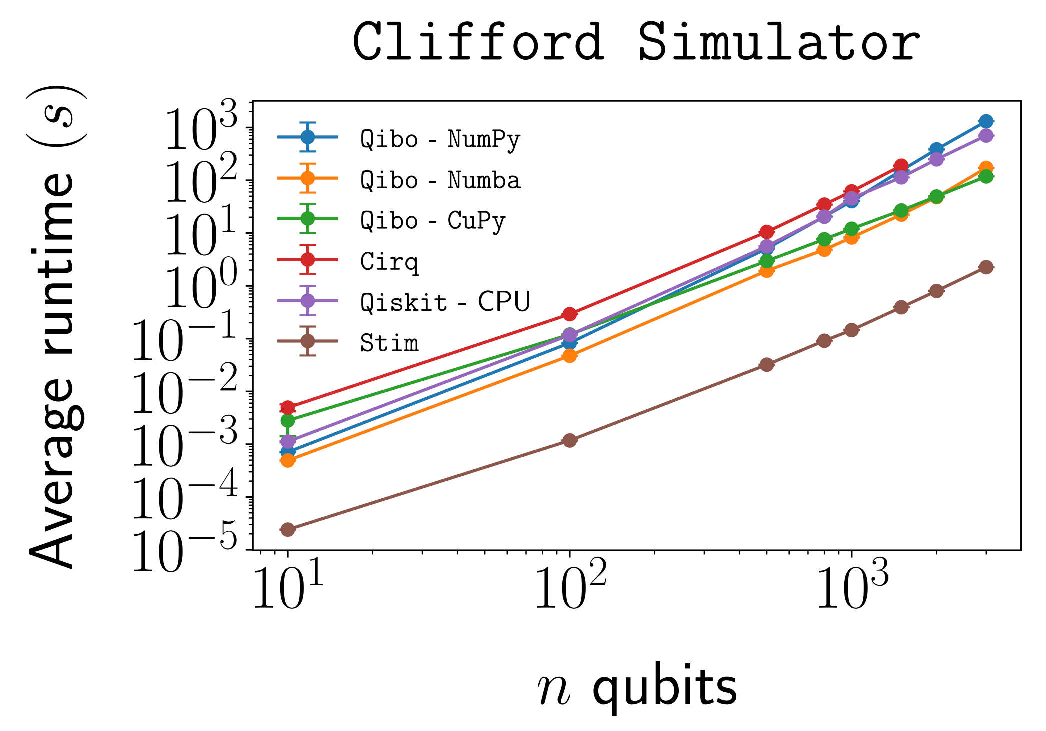
$$\left(\begin{array}{ccc|ccc|c} x_{11} & \cdots & x_{1n} & z_{11} & \cdots & z_{1n} & r_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{n1} & \cdots & x_{nn} & z_{n1} & \cdots & z_{nn} & r_n \\ \hline x_{(n+1)1} & \cdots & x_{(n+1)n} & z_{(n+1)1} & \cdots & z_{(n+1)n} & r_{n+1}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

Instead of operating on the whole state vector, the state is represented by a much more compressed *tableau*.

It still requires vectorized operations on the boolean entries, that can be optimized in a similar fashion to the general state vector approach.

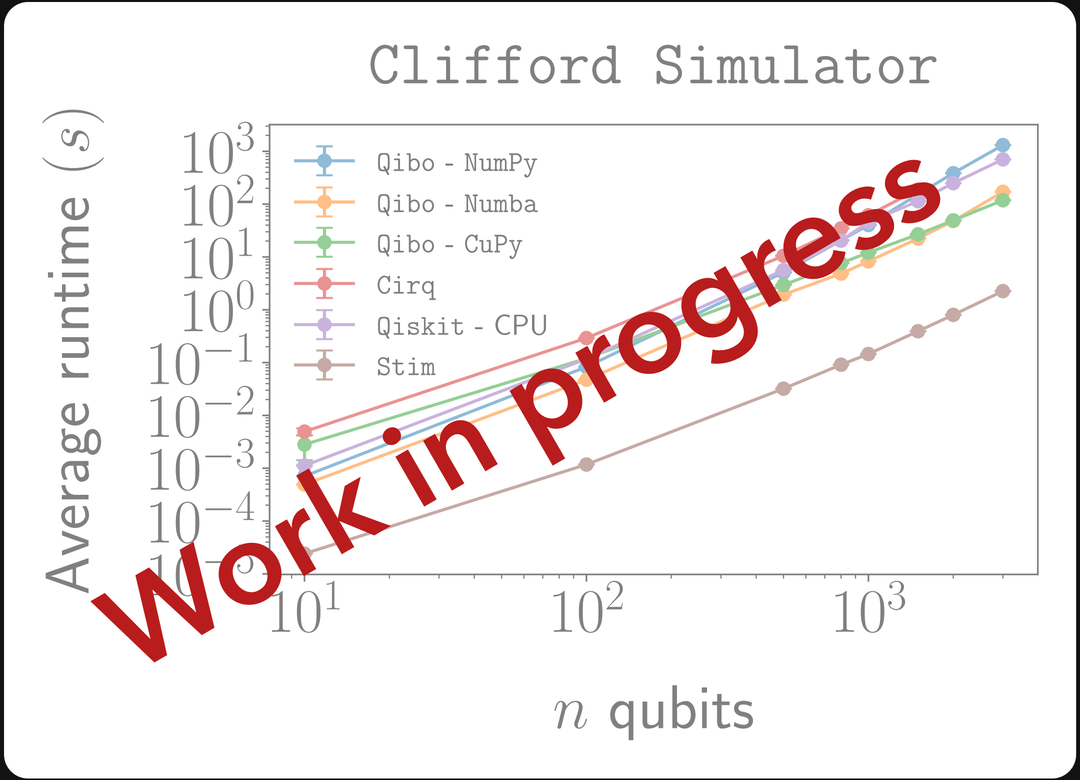
Clifford

Benchmarks



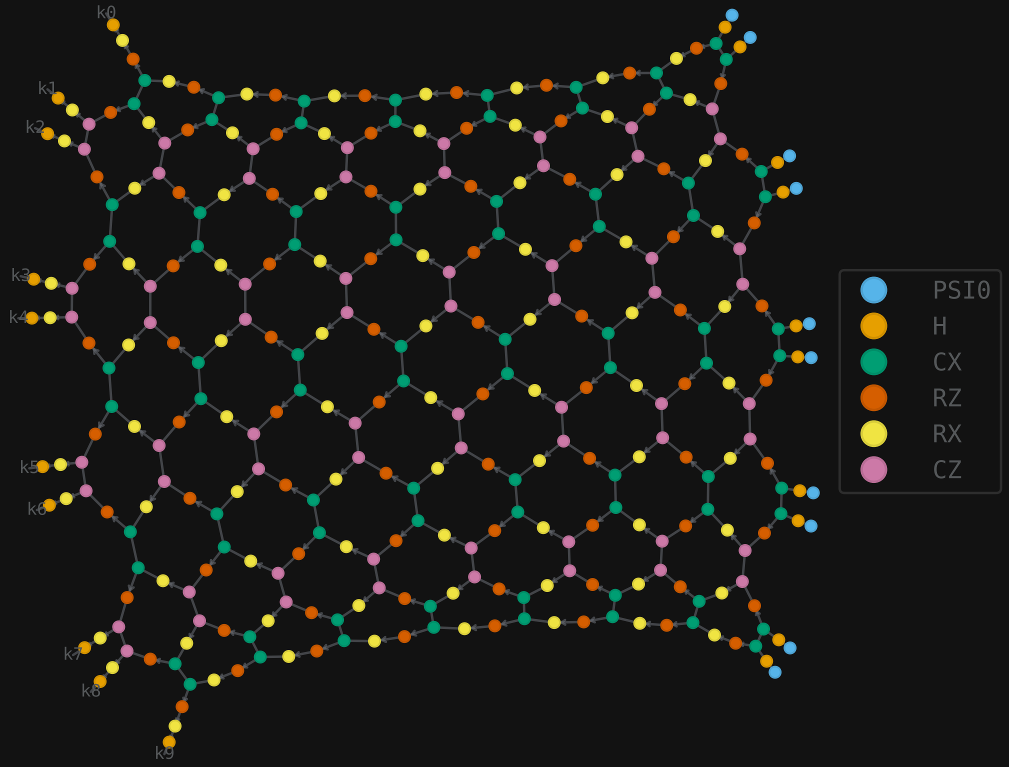
Clifford

Benchmarks

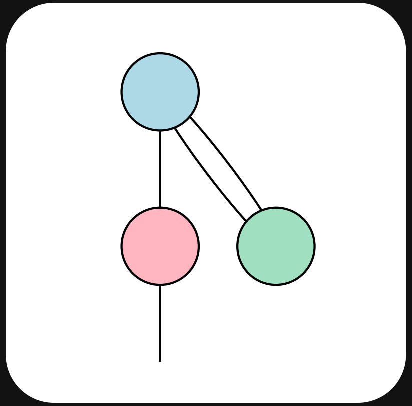


Tensor network

Optimized for observables.

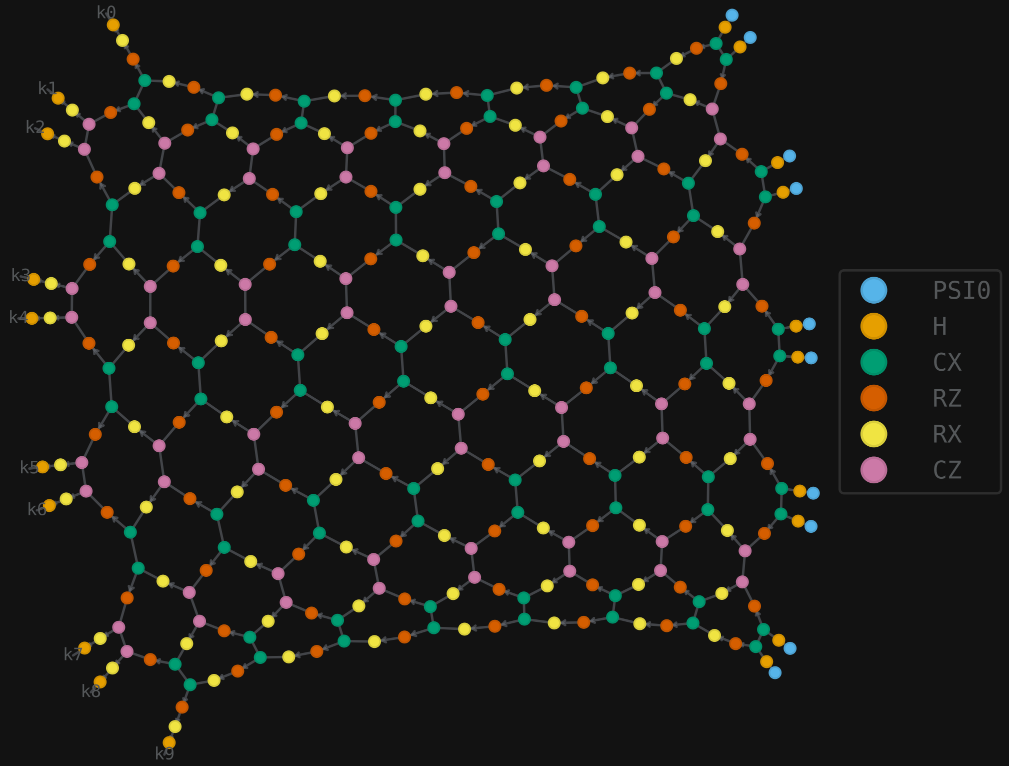


Contractions

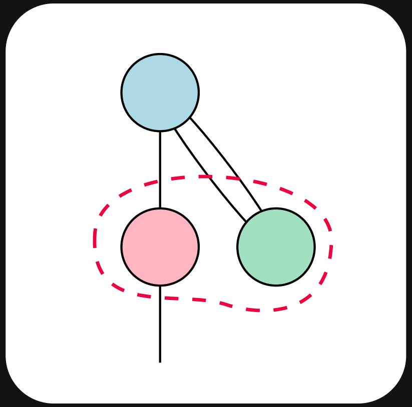


Tensor network

Optimized for observables.

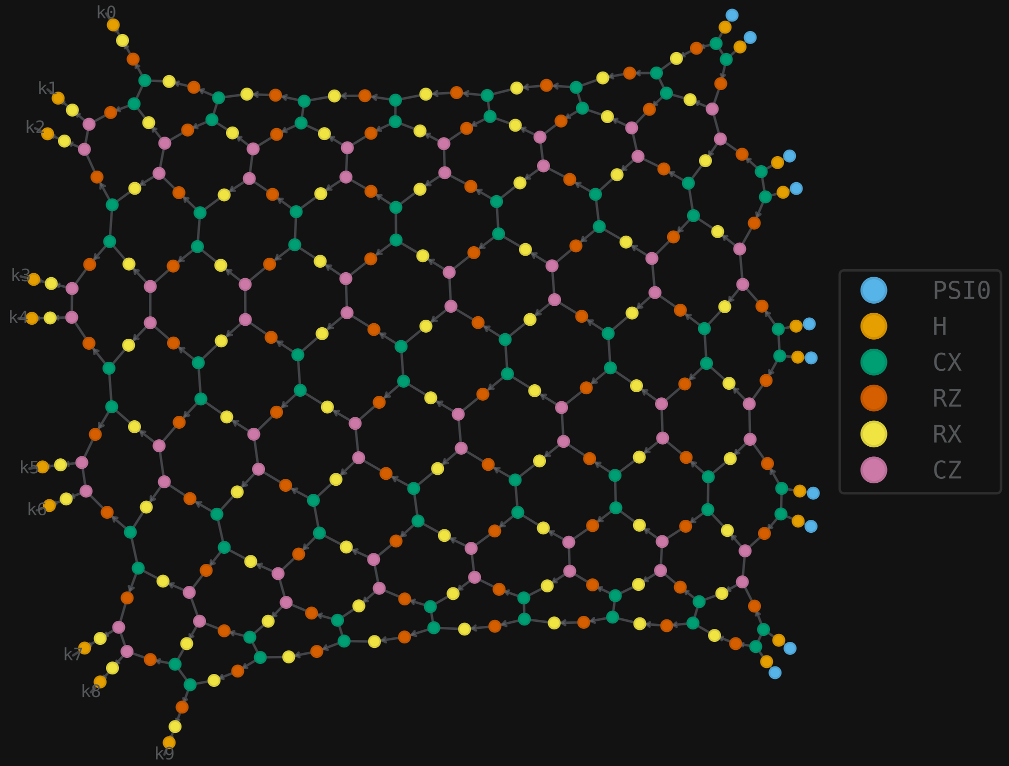


Contractions

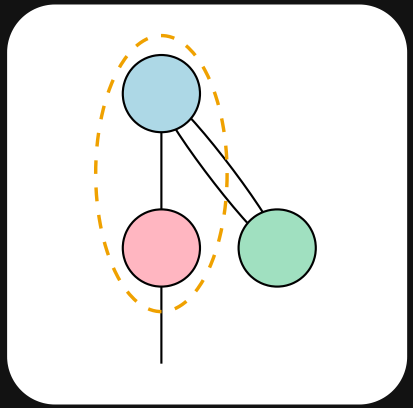


Tensor network

Optimized for observables.

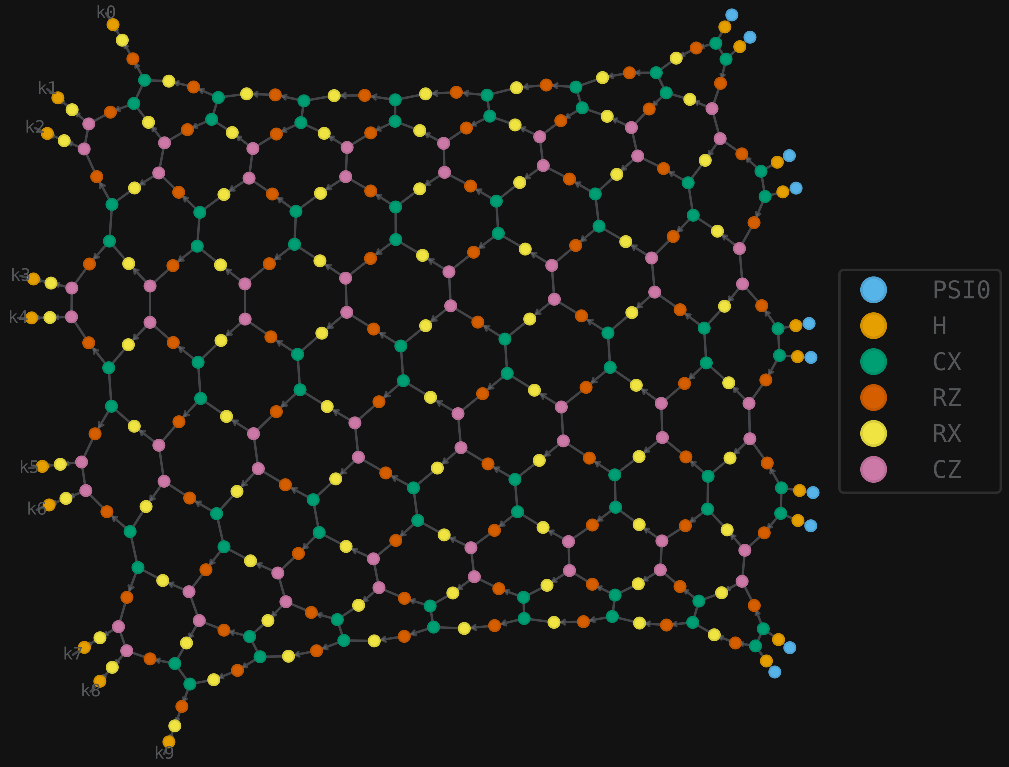


Contractions

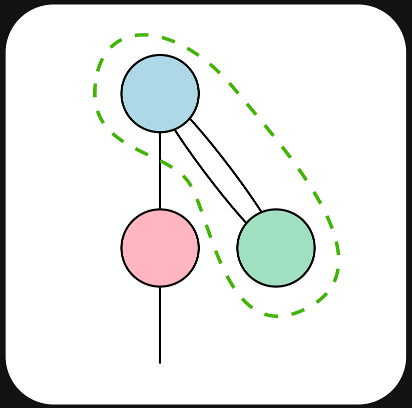


Tensor network

Optimized for observables.



Contractions

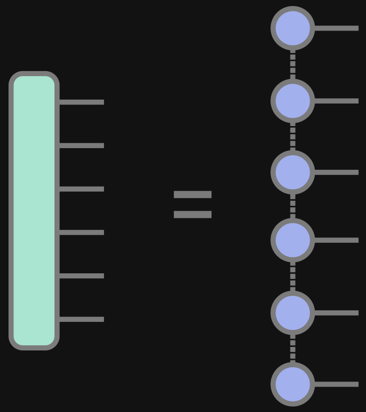


Tensor network

beyond `opt_einsum`

Approximation

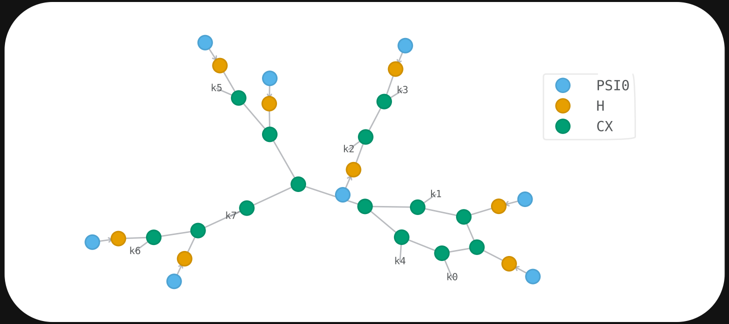
Based on singular value decomposition (SVD).



A very frequent matrix product state (MPS).

But also other ansatzes are used.

Workload distribution



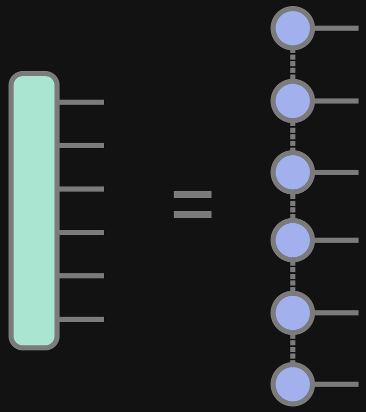

```
for q in range(nq):
    c.apply_gate('H', q)

for q in range(0, nq, 2):
    c.apply_gate('CNOT', q, q + 1)

c.apply_gate('CNOT', 4, 7)
c.apply_gate('CNOT', 4, 1)
c.apply_gate('CNOT', 4, 0)
```

Tensor network

beyond `opt_einsum`

Approximation

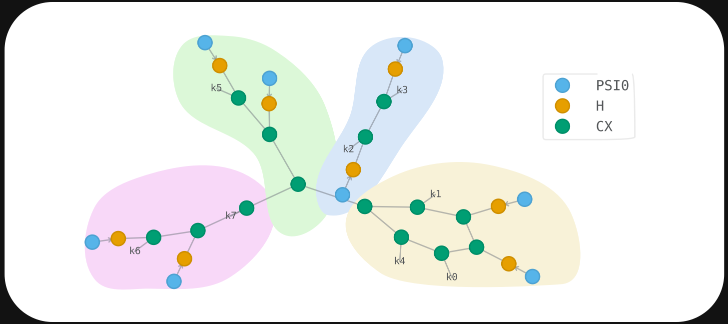
Based on singular value decomposition (SVD).



A very frequent matrix product state (MPS).

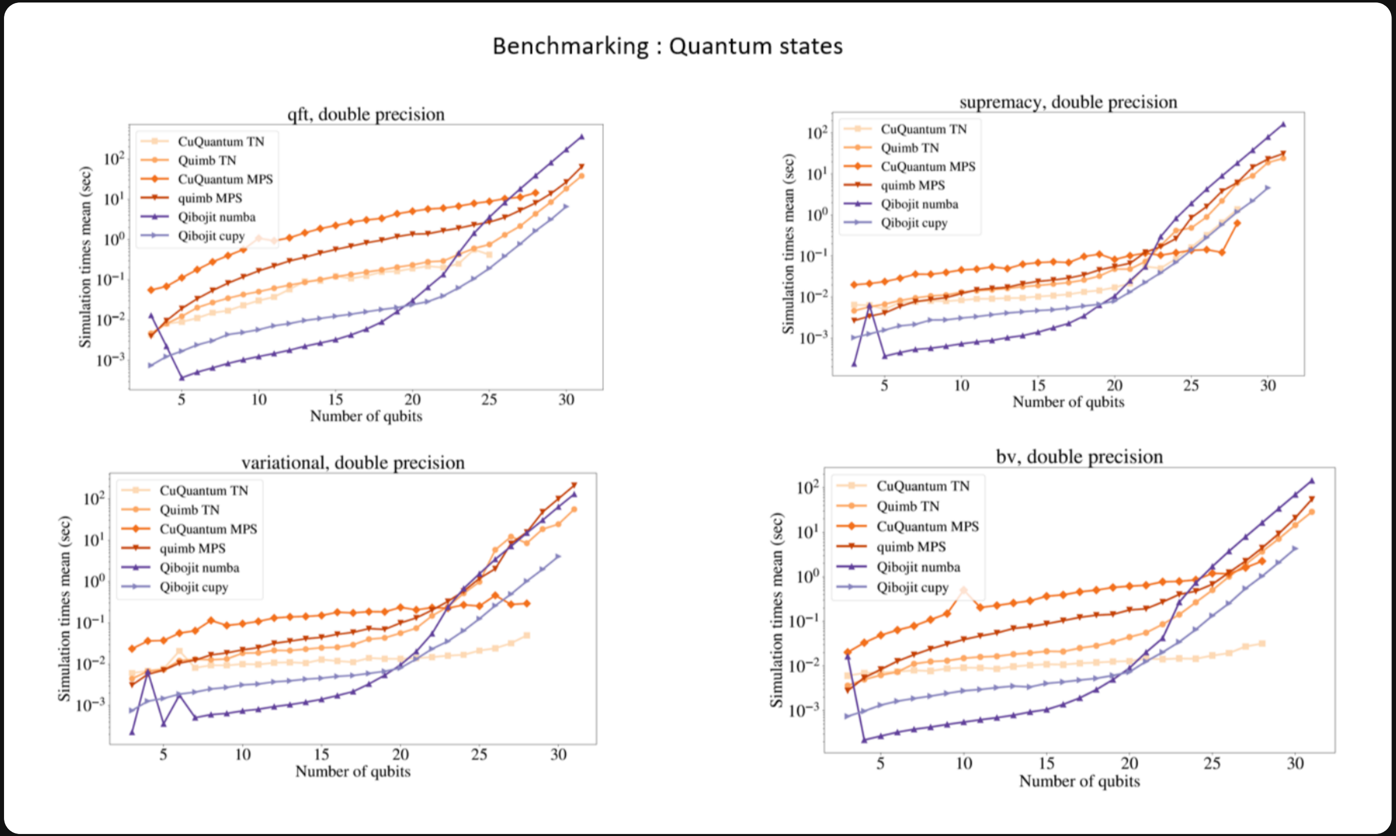
But also other ansatzes are used.

Workload distribution




```python
for q in range(nq):
    c.apply_gate('H', q)

for q in range(0, nq, 2):
    c.apply_gate('CNOT', q, q + 1)

c.apply_gate('CNOT', 4, 7)
c.apply_gate('CNOT', 4, 1)
c.apply_gate('CNOT', 4, 0)
```

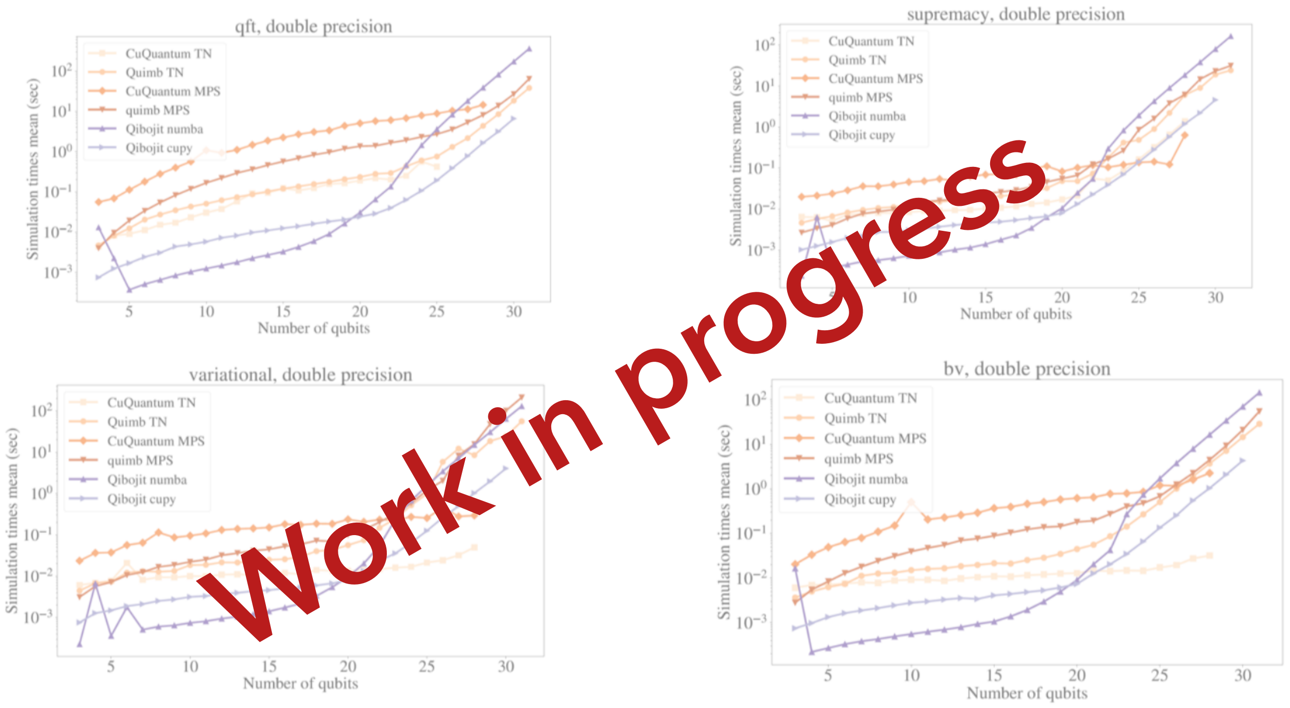
QiboTN



Benchmarking : Quantum states
qft, double precision
supremacy, double precision
variational, double precision
bv, double precision
CuQuantum TN
Quimb TN
CuQuantum MPS
quimb MPS
Qibojit numba
Qibojit cupy
Simulation times mean (sec)
Number of qubits

QiboTN

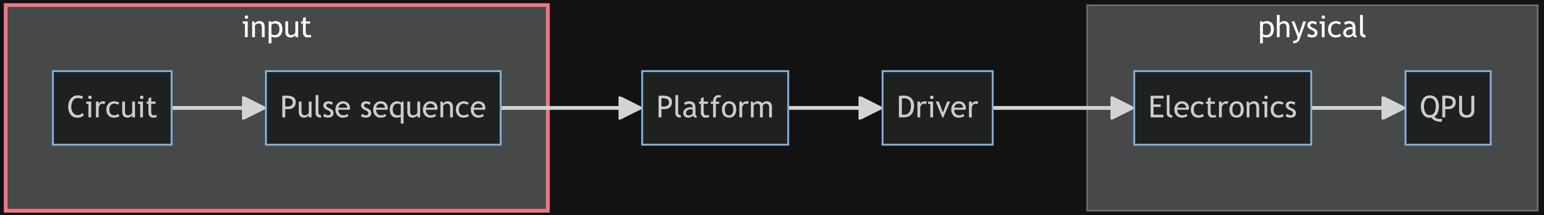
Benchmarking : Quantum states



Qibolab `[arXiv: 2308.06313]`

Quantum control

Qibolab - Interface



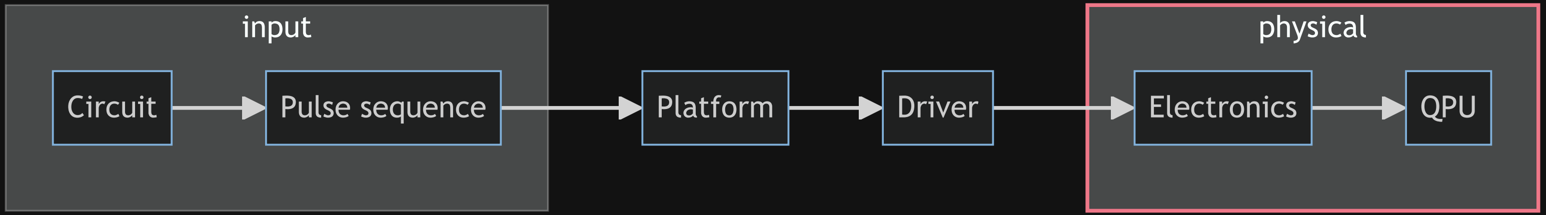
The **input** for a computation could be very standard, at the level of a **circuit**. That kind of interface is already defined by Qibo itself.

However, at a lower level, **pulses** are still a standard-enough way to interact with hardware, and these are defined by Qibolab.

Pulse sequence plot (from notebook?)

```
def create():
    instrument = DummyInstrument("myinstr", "0.0.0.0:0")

    channels = ChannelMap()
    channels |= Channel(
        "readout",
        port=instrument.ports("o1")
    )
    ...

    return Platform(
        "myplatform",
        qubits={qubit.name: qubit},
        instruments={instrument.name: instrument},
        ...
    )
```

Qibolab - Drivers

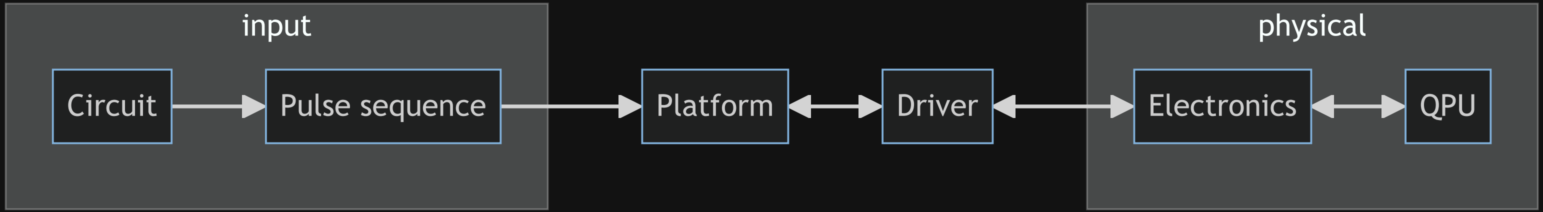


- Qblox
- Zurich
- QM
- QICK

```
      move      1,R0        # Start at marker output channel 0 (move 1 into R0)
      nop                   # Wait a cycle for R0 to be available.

loop: set_mrk   R0          # Set marker output channels to R0
      upd_param 1000        # Update marker output channels and wait 1μs.
      asl       R0,1,R0     # Move to next marker output channel (left-shift R0).
      nop                   # Wait a cycle for R0 to be available.
      jlt       R0,16,@loop # Loop until all 4 marker output channels have been set once.

      set_mrk   0           # Reset marker output channels.
      upd_param 4           # Update marker output channels.
      stop                  # Stop sequencer.
```

by Qblox

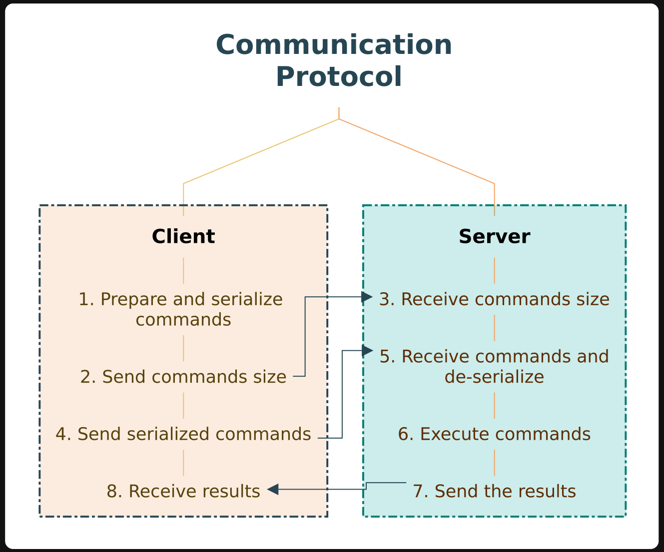
Qibosoq - Server on QICK [arXiv: 2310.05851]



Qibolab handles the whole connection, and takes care of fetching the single or multiple results.

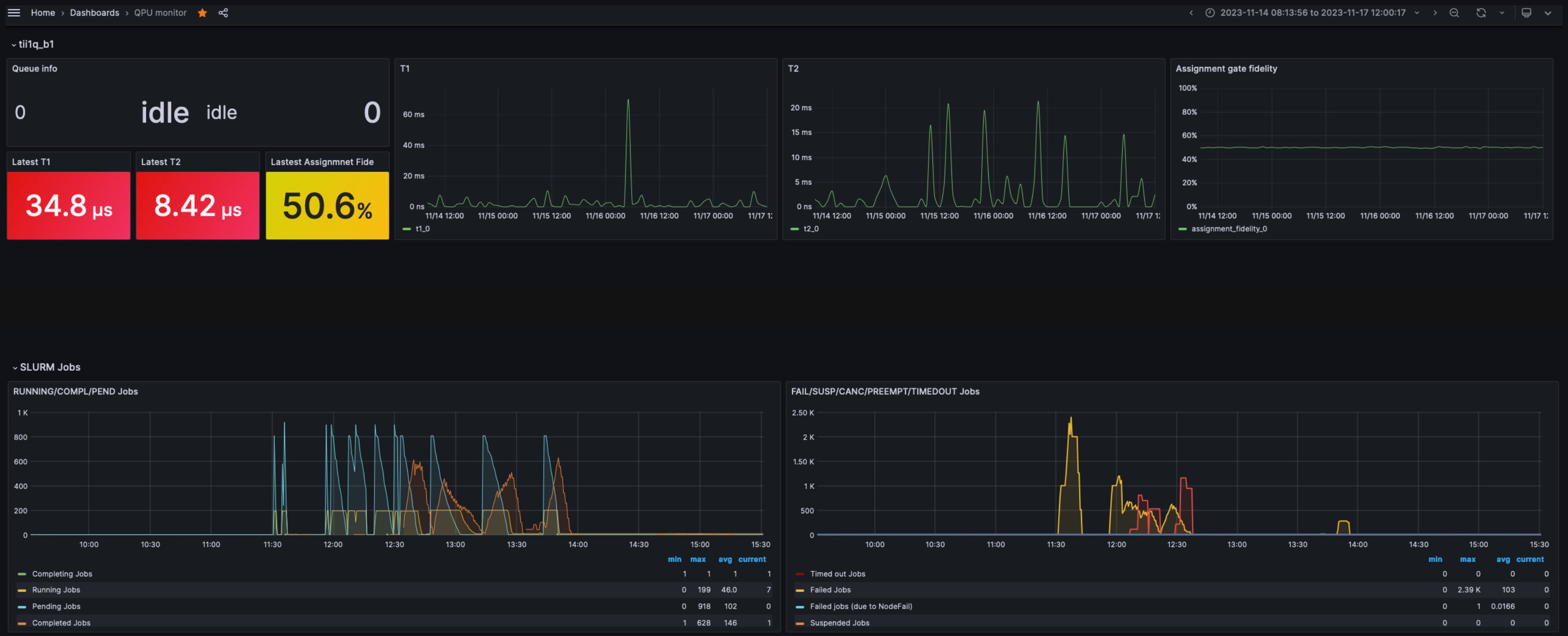
For the single open source platform *FPGA FIRMWARE* currently in Qibolab, there has been a dedicate effort to define a suitable server, to optimize the communication with the board.

⟶ Qibosoq



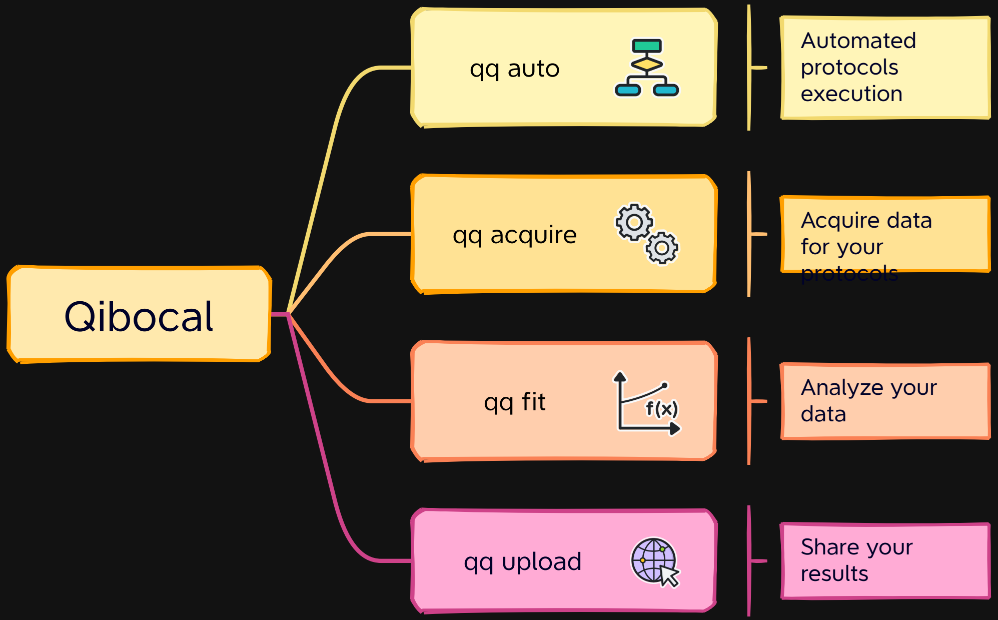
in collaboration with ***INFN-UNIMIB-BIQUTE***

Platform dashboard



Qibocal

An owed mention



Qibocal Reports

Uploaded Reports

Please select a report from the table below:

Search:

Title	Date	Platform	Start-time (UTC)	End-time (UTC)	Tag	Author
test_tii1qb1	2024-02-13	/home/users/andrea.pasquale/qibolab_platforms_qrc/tii1qs_xld1000	06:53:18	06:53:26	-	andrea.pasquale
test_qubit_spec_tii1qs	2024-02-13	/home/users/andrea.pasquale/qibolab_platforms_qrc/tii1qs_xld1000	06:59:45	06:59:50	-	andrea.pasquale
web_calibration_report_20240209_163420	2024-02-09	/home/users/qibocal/webapp/qibolab_platforms_qrc/iqm5q	12:34:25	12:34:51	web_calibration	qibocal
web_calibration_report_20240209_154537	2024-02-09	/home/users/qibocal/webapp/qibolab_platforms_qrc/iqm5q	11:45:54	11:46:21	web_calibration	qibocal
web_calibration_report_20240209_163420	2024-02-09	/home/users/qibocal/webapp/qibolab_platforms_qrc/iqm5q	12:34:25	12:34:51	web_calibration	qibocal

iqm5q/calibration_november/10112023/qutrit

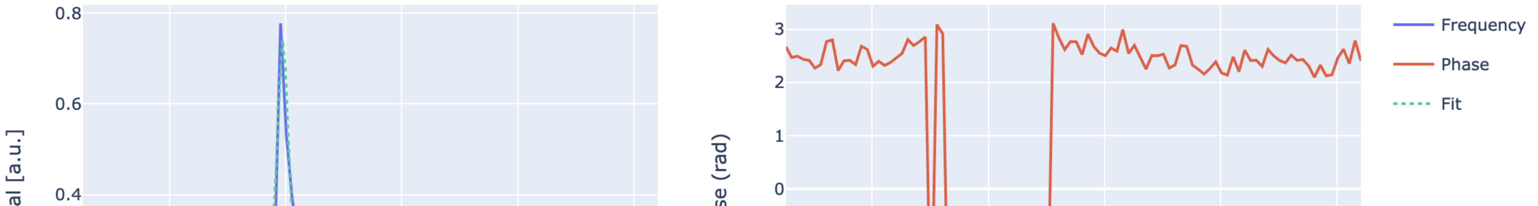
Export to pdf

Platform: IQM5q
Run date: 2023-11-10
Start time (UTC): 15:42:15
End time (UTC): 15:42:15

Qubit Spectroscopy 01 - 0

- Qubit 0

Qubit	Parameters	Values
0	qubit frequency	4079523906.0
0	amplitude	0.5



Thanks

Hybrid compilation

CATALYST BETA

Target both simulation and hardware. Optimize classical and quantum instructions. Optimal for QML-like hybrid applications.

Continuous variables

Simulation

Strawberry Fields