

Towards a hybrid quantum operating system

First Year PhD students workshop

Andrea Pasquale

29th September 2022



UNIVERSITÀ
DEGLI STUDI
DI MILANO

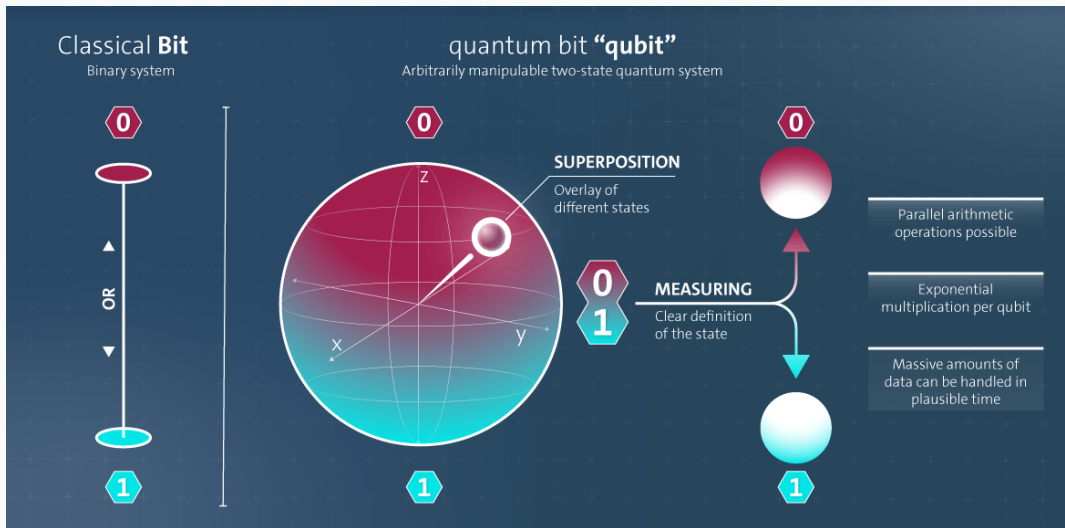




- Chips with 1 and 5 qubits at TII
- Chips with 1 and 2 qubits at Qilimanjaro
- Chip with 1 qubit in Italy (soon)
- Chips with up to 10 qubits at CQT

Introduction on Quantum computing

What is a Qubit?



Qubit

A qubit is a two state quantum system

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle ,$$

where α and β are complex numbers such that $|\alpha|^2 + |\beta|^2 = 1$.

Quantum circuits

According to Quantum Mechanics (QM) a state ψ evolves through **unitary** operators U_i

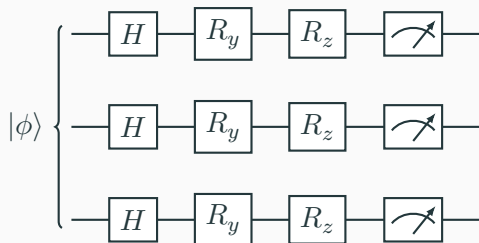
$$|\psi\rangle \rightarrow |\psi'\rangle = U_2 U_1 |\psi\rangle$$

In the gates-based model we obtain a circuit:



which we call **quantum circuit**.

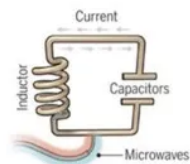
With multiple qubits we get the following:



A n qubits system is described by a vector state $|\phi\rangle$ with 2^n components.

How can we implement physical qubits?

Quantum technologies

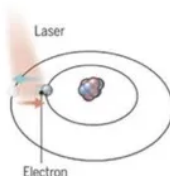


Superconducting loops

A resistance-free current oscillates back and forth around a circuit loop. An injected microwave signal excites the current into superposition states.

Longevity (seconds)
0.00005

Logic success rate
99.4%



Trapped ions

Electrically charged atoms, or ions, have quantum energies that depend on the location of electrons. Tuned lasers cool and trap the ions, and put them in superposition states.

>1000

99.9%

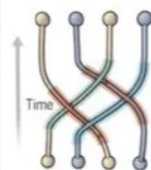


Silicon quantum dots

These "artificial atoms" are made by adding an electron to a small piece of pure silicon. Microwaves control the electron's quantum state.

0.03

~99%

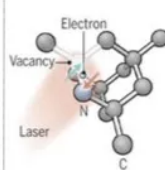


Topological qubits

Quasiparticles can be seen in the behavior of electrons channeled through semiconductor structures. Their braided paths can encode quantum information.

N/A

N/A



Diamond vacancies

A nitrogen atom and a vacancy add an electron to a diamond lattice. Its quantum spin state along with those of nearby carbon nuclei, can be controlled with light.

10

99.2%

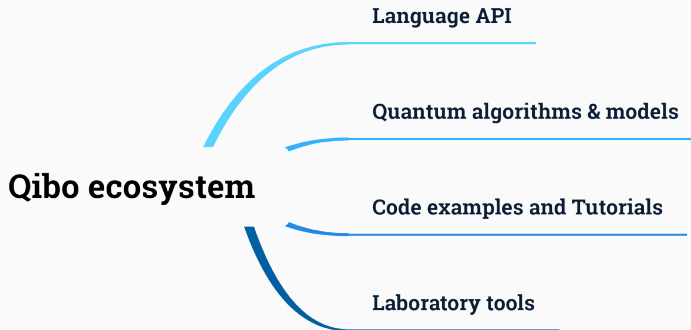


Source IBM

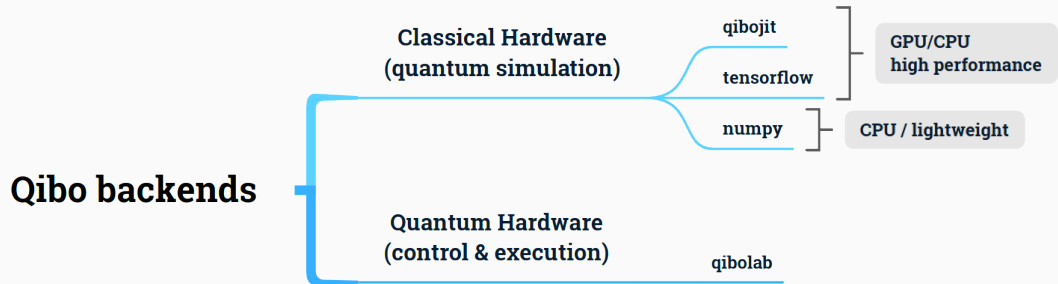
Quantum Technologies | Sample | www.yole.fr | ©2020

Introducing Qibo

Qibo is an **open-source** full stack API for quantum simulation and quantum hardware control and calibration.



A modular framework for quantum computing



Introducing Qibojit

Matrix multiplication to simulate circuits:

$$\psi'(\sigma_1, \dots, \sigma_n) = \sum_{\tau'} G(\tau, \tau') \psi(\sigma_1, \dots, \tau', \dots, \sigma_n) .$$

✗ Number of operations scales **exponentially** with the number of qubits!

We need more sophisticated backends to perform simulation:

✗ NumpyBackend : **Numpy** tensors and primitives

✗ TensorFlowBackend : **Tensorflow** tensors and primitives

✓ QibojitBackend : Just-In-time

</> CPU : **Numpy** tensor + custom operations with **Numba JIT**

</> GPU(S) : **Cupy** tensors + custom operations using

- **Cupy JIT** Raw kernels
- **NVIDIA cuQuantum** API

Paper published on Quantum:

<https://quantum-journal.org/papers/q-2022-09-22-814/>

```
from numba import njit, prange

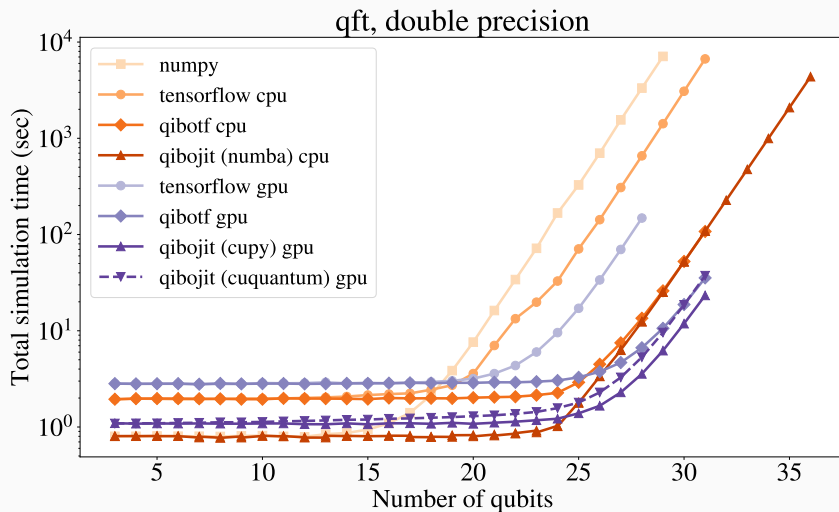
@njit(parallel=True, cache=True)
def apply_gate_kernel(state, gate, target):
    """Operator that applies an arbitrary one-qubit gate.

    Args:
        state (np.ndarray): State vector of size (2 **
        ↪ nqubits,).
        gate (np.ndarray): Gate matrix of size (2, 2).
        target (int): Index of the target qubit.
    """
    k = 1 << target
    # for one target qubit: loop over half states
    nstates = len(state) // 2
    for g in prange(nstates):
        # generate index with fast binary operations
        i1 = ((g >> m) << (m + 1)) + (g & (k - 1))
        i2 = i1 + k
        state[i1], state[i2] = (gate[0, 0] * state[i1] + \
                                gate[0, 1] * state[i2],
                                gate[1, 0] * state[i1] + \
                                gate[1, 1] * state[i2])

    return state
```

To further speed up:

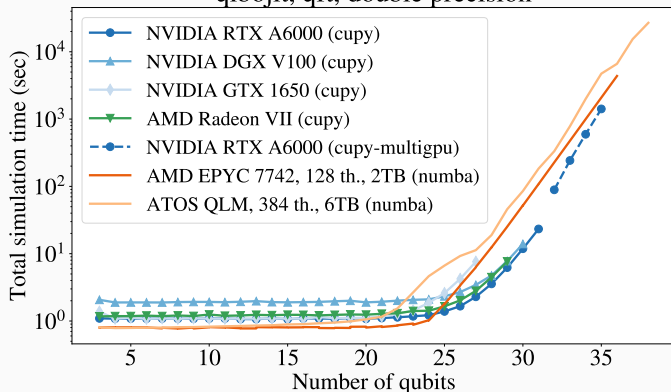
- *in-place updates*
- exploit sparsity of matrices
- specialized operators for:
 - single qubit gate: X, Y Z
 - two qubit gates: SWAP



Benchmark library: <https://github.com/qiboteam/qibojit-benchmarks>

Benchmarks

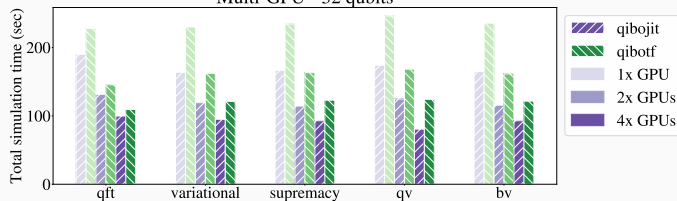
qibojit, qft, double precision



Qibojit features

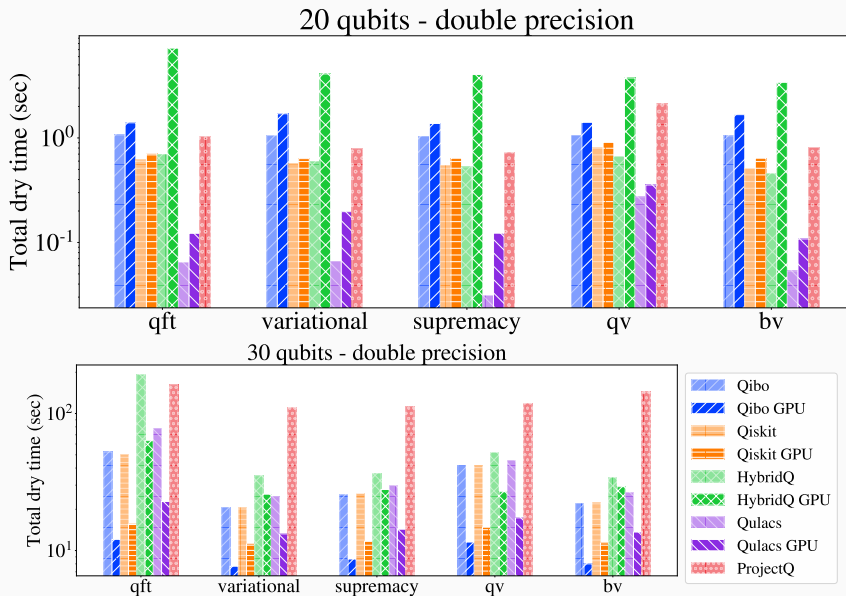
- Support for CPU, GPU and multi-GPU
- NVIDIA and AMD (ROCm) GPUs
- Reduced memory footprint

Multi-GPU - 32 qubits



Benchmark library: <https://github.com/qiboteam/qibojit-benchmarks>

How does Qibo perform against the other libraries?

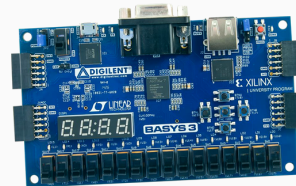
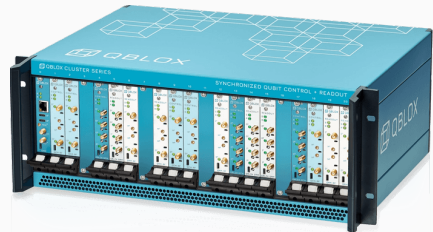


Benchmark library: <https://github.com/qiboteam/qibojit-benchmarks>

Hardware control using Qibo

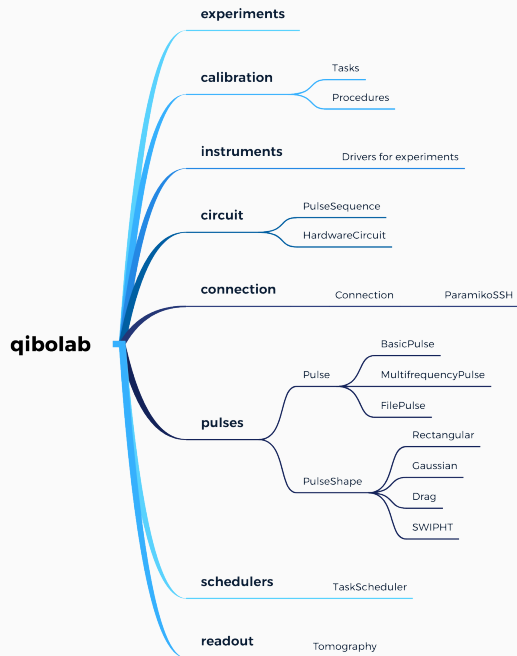
Hardware control

For superconducting qubits **gates** are implemented by sending **pulses**.



We need a framework to control all these devices at the same time.

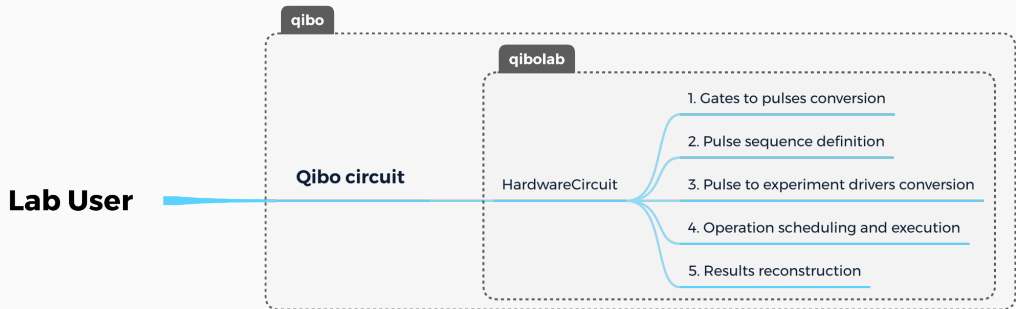
Introducing Qibolab



Qibolab features:

- Deploy Qibo models on quantum hardware easily
- User-friendly Pulse API
- Create custom experimental drivers for lab setup
- Support multiple heterogeneous platforms

How to use qibolab?



```
from qibo import models, gates

circuit = models.Circuit(nqubits=1)
circuit.add(gates.H(0))
circuit.add(gates.X(0))
circuit.add(gates.M(0))

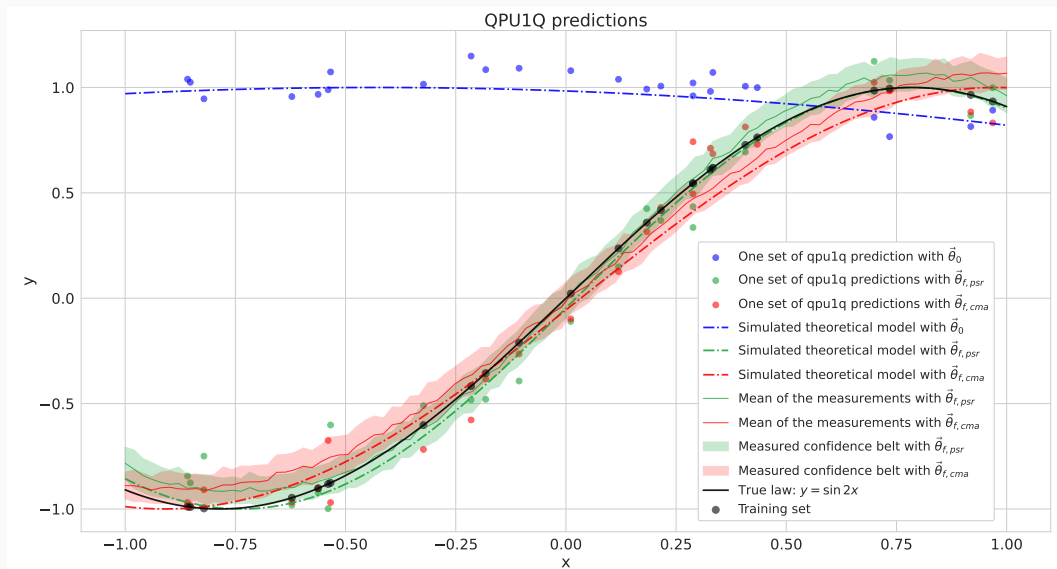
# Simulate the circuit
set_backend("qibojit")
simulation = circuit()

# Execute circuit on quantum hardware
set_backend("qibolab")
hardware = circuit()
```

- A single object to execute both on hardware and simulation
- Job scheduling to access the hardware using slurm



Quantum Machine Learning on Real Hardware



A reporting tool for calibration using Qibo

Difficulties

Suppose that we have assembled a quantum computer and we have a way to send pulses to the chip... are we done? **No**

We need to **characterize**, **validate** and **verificate** our qubits (QCVV):

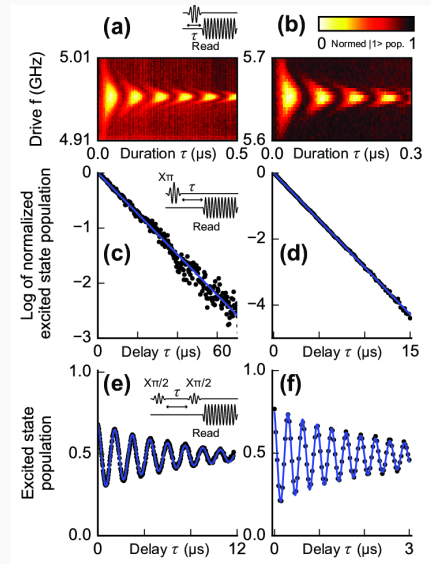
▶ Perform standard calibration routines:

- 🔧 Resonator and qubit spectroscopy
- 🔧 Rabi and Ramsey
- 🔧 T1 and T2 determination

▶ Perform quantum protocols to extract the fidelity:

- 🔧 Randomized Benchmarking
- 🔧 Gate Set Tomography
- 🔧 Cross-Entropy Benchmarking

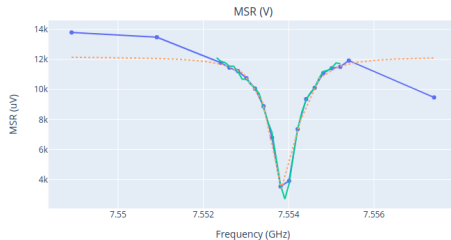
▶ Repeat the above steps periodically.



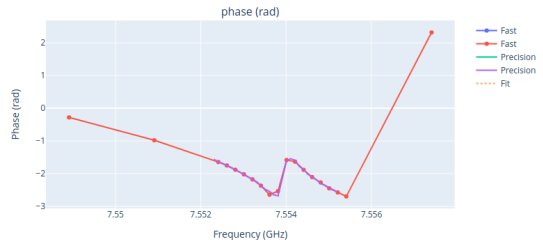
A new reporting tool for Qibo

We are developing a new tool that it will be able to perform QCVV in Qibo with the following features:

- ▶ Platform agnostic
- ▶ Launch calibration routine easily
- ▶ Live-plotting tools
- ▶ Live-fitting tools
- ▶ Save and share your data
- ▶ Autocalibration routines



The estimated resonator_freq is 7553897.4 Hz.
The estimated peak_voltage is 3105.718 uV.



QCVV Reports

Home

Timestamp
Summary

Actions

Resonator Punchout

- Frequency vs Attenuation
- MSR vs Frequency

Qubit Spectroscopy

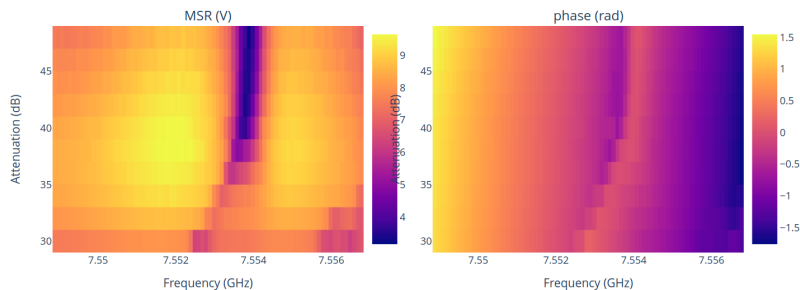
- MSR and Phase vs Frequency

Rabi Pulse Amplitude And
Attenuation

- MSR vs length and amplitude

Resonator Punchout

Frequency vs Attenuation - Qubit 1



Outlook

Qibo is growing to accomodate different tasks:

- ✓ High-performance quantum simulation: [qibojit](#)
- ✓ Hardware control: [qibolab](#)
- ✓ Hardware calibration: [qcvv](#)



What makes Qibo different from other libraries:

- + Public available as an open source project.
- + Modular layout design with possibility of adding
 - a new backend for simulation
 - a new platform for hardware control
- + Community driven effort

<https://github.com/qiboteam/qibo>

<https://qibo.readthedocs.io/en/stable/>

Thanks for listening!
