Towards an automatized framework to perform quantum calibration

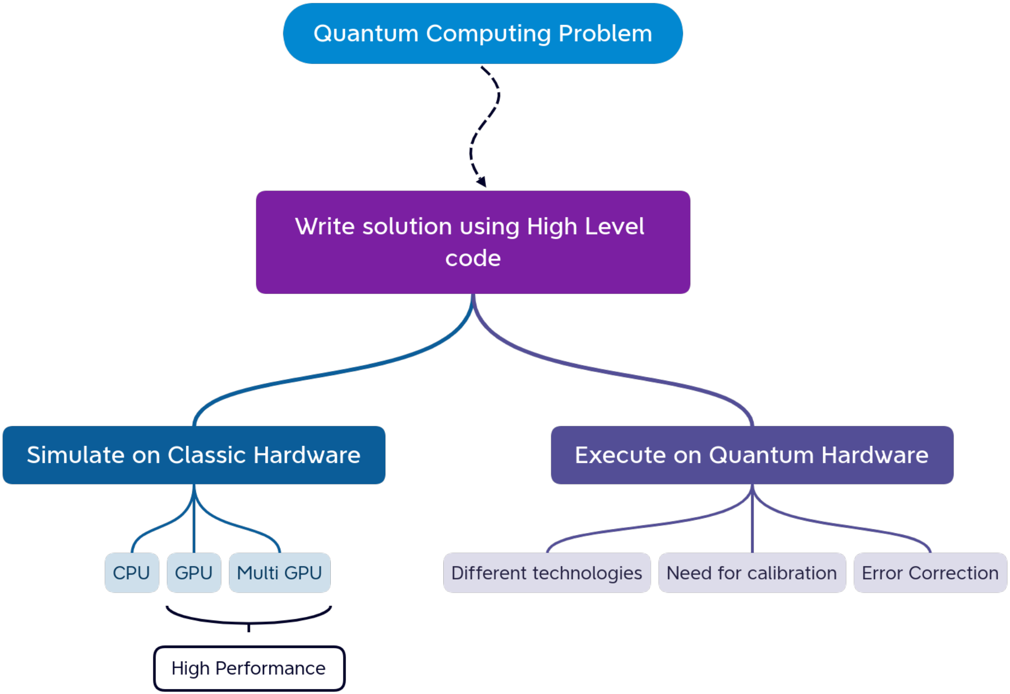
An introduction to Qibo

Andrea Pasquale, on the behalf of the Qibo collaboration

27th October 2022, ACAT 2022, Bari

Institute	TII	CQT	INFN	Qilimajiaro
Quantum Hardware	5 qubits	10 qubits	1 qubit	2 qubits

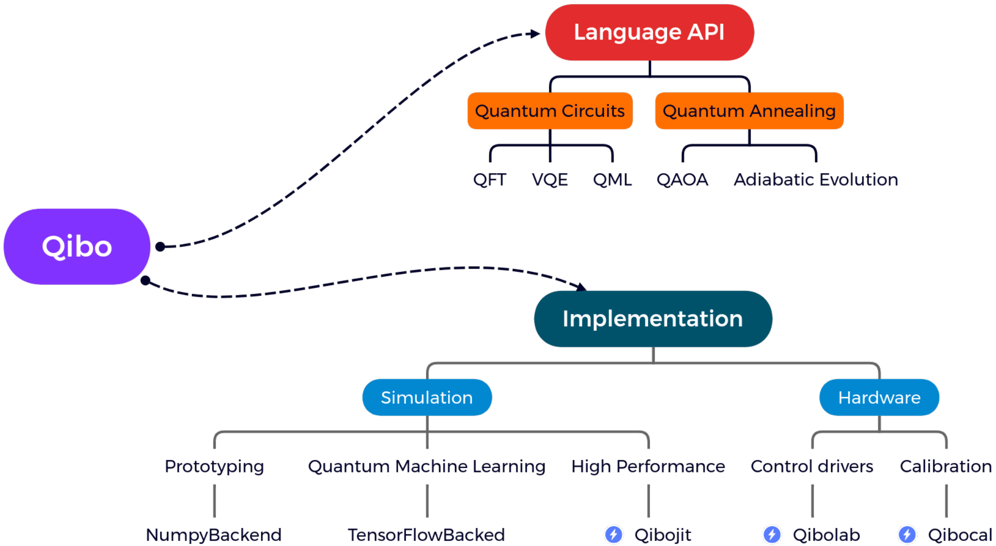
Challenge



Is to possible to create from scratch a framework for all of this?

Introducing Qibo

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



`https://github.com/qiboteam/qibo`

Introducing Qibojit

Matrix multiplication to simulate circuits:

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

✖ 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/`

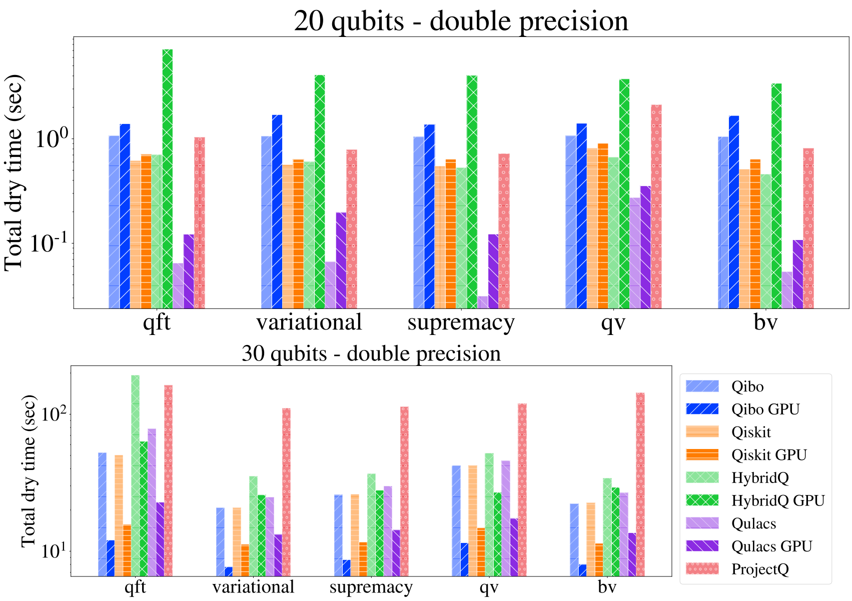
Qibojit - Example

```python
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(states) // 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

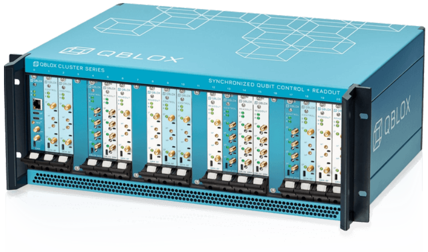
How does Qibo perform against the other libraries?

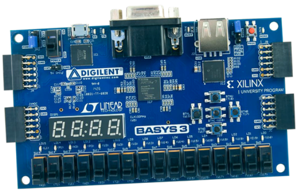


Benchmark library: `https://github.com/qiboteam/qibojit-benchmarks`

Hardware control using Qibo

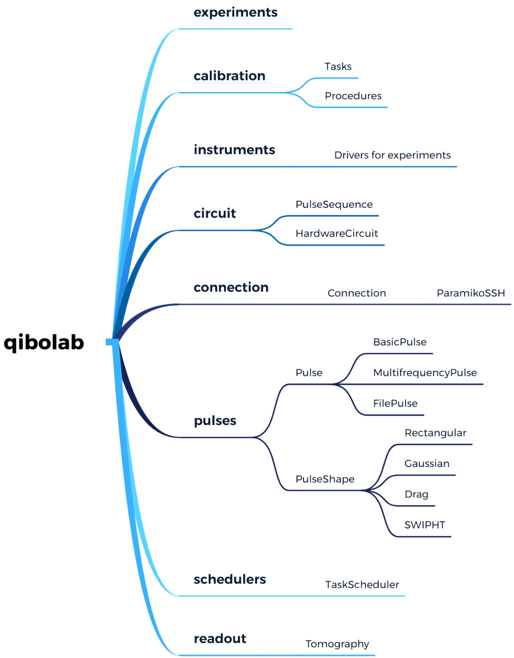
For superconducting qubits **gates** are implemented by sending **pulses**.





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

Introducing Qibolab



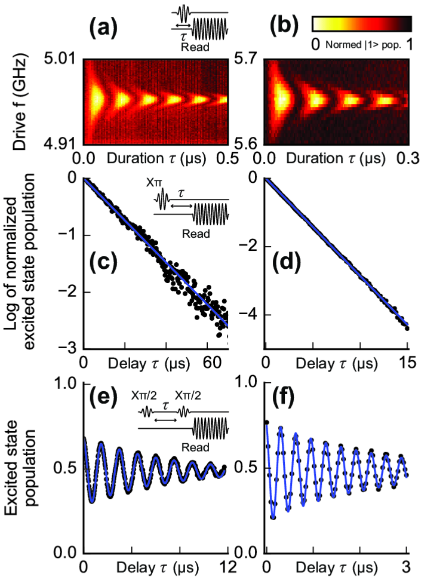
Qibolab key features:

- Create **custom experimental drivers** for lab setup
- Platform **agnostic** layout
- Deploy **Qibo models** on quantum hardware easily

A reporting tool for calibration using Qibo

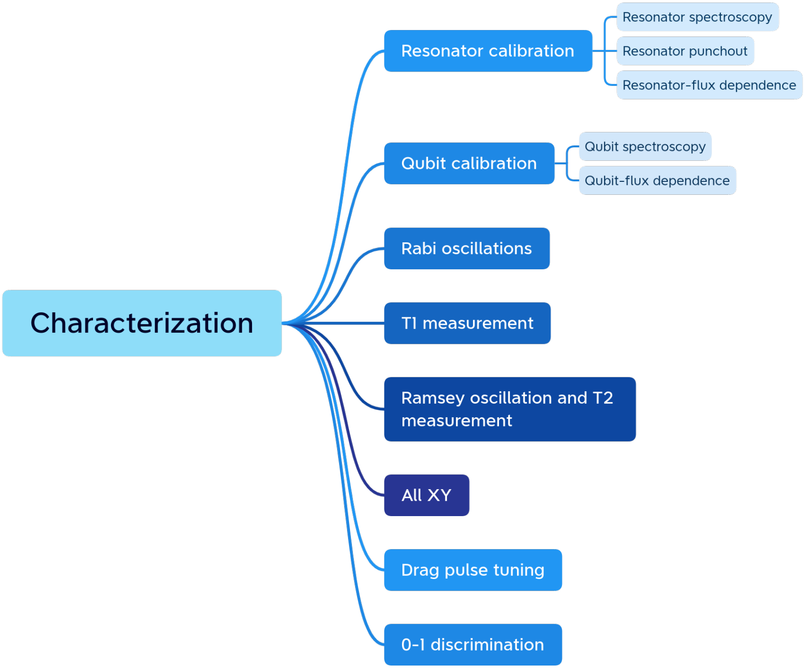
Suppose that we have assembled a quantum computer and we have a way to send pulses to the chip... are we done? **No**

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



	Q0	Q1	Q2	Q3	Q4
Frequency (GHz)	5.28	5.21	5.02	5.28	5.07
T1 (μs)	62.40	55.10	48.40	59.00	53.30
T2 (μs)	77.50	64.00	54.70	57.30	36.40
Gate error (10^{-3})	1.37	1.37	2.23	1.72	0.94
Readout error (10^{-2})	2.40	2.60	3.00	2.20	4.50
	CX0_1	CX1_2		CX3_2	CX4_2
MultiQubit gate error (10^{-2})	2.72	3.77		3.97	3.51
	CX0_2			CX3_4	
	4.18			3.62	

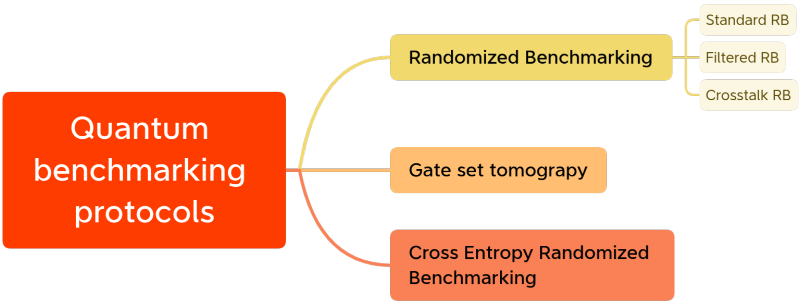
Single Qubit characterization



Quantum benchmarking protocols

After characterizing our quantum hardware we need to compute the gates error behavior.

In the current state-of-the-art this is computed using **Quantum benchmarking protocols**.



In Qibocal we are currently developing a suite for the execution of the latest QBP available.

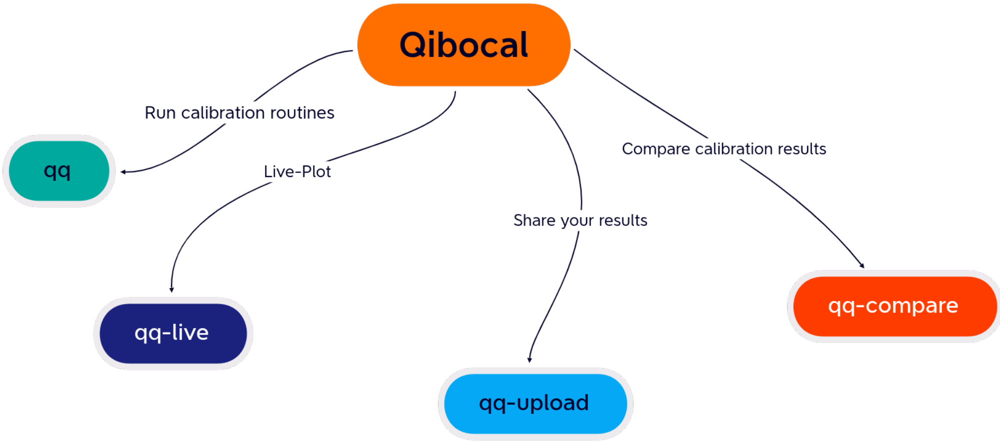
Motivation

We are developing a new tool called **Qibocal** to perform qubits calibration in Qibo using Qibolab as the main driver.

The main features that we are implemented are the following:

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

Qibocal: implementation



How to use qq

To run a specific set of calibration it is sufficient to write a runcard:

```yaml
platform: tii5q

qubits: [2]

format: csv

actions:
  resonator_spectroscopy:
    lowres_width: 5_000_000
    lowres_step: 2_000_000
    highres_width: 1_500_000
    highres_step: 200_000
    precision_width: 1_500_000
    precision_step: 100_000
    software_averages: 1
    points: 1

  qubit_spectroscopy:
    fast_start: -50_000_000
    fast_end: 50_000_000
    fast_step: 500_000
    precision_start: -500_000
    precision_end: 500_000
    precision_step: 100_000
    software_averages: 1
    points: 1
```

You can execute the following runcard by typing:

```
qq <runcard.yaml>
```

qq will take care of:

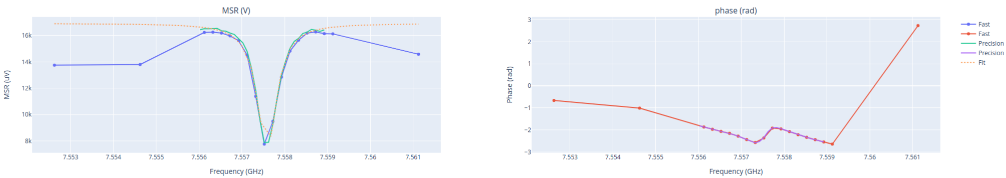
- connecting to the platform
- executing the routines listed under actions
- generating an update runcard for the platform
- generating a web report containing the results

How to use qq-live

Using qq-live it is possible to visualize the results during (after) the execution

Resonator Spectroscopy

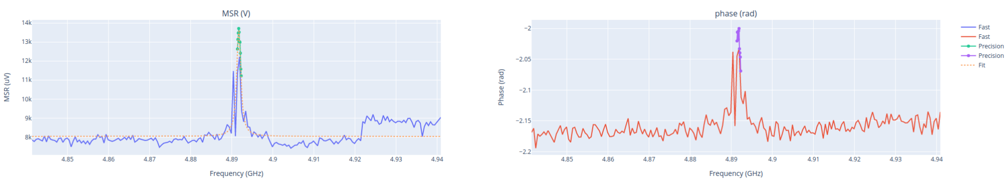
MSR and Phase vs Frequency - Qubit 2



The estimated resonator_freq is 7557586763.3 Hz.
The estimated peak_voltage is 7372.400 uV.

Qubit Spectroscopy

MSR and Phase vs Frequency - Qubit 2



The estimated qubit_freq is 4891680320.9 Hz.
The estimated peak_voltage is 13687.512 uV.

How to use qq-upload

You can share your results by uploading the report generated by qq using `qq-upload`

Qibocal Reports

- **Home**
  - Timestamp
  - Summary
- **Actions**
  - Resonator Spectroscopy
    - MSR and Phase vs Frequency
  - Rabi Pulse Length
    - MSR vs Time

2022-10-18-000-andrea-pasquale

Platform: tii5q
Run date: 2022-10-18
Start time (UTC): 08:43:01
End time (UTC): 08:43:59

Summary

In the table below we show the libraries and respective versions used in 2022-10-18-000-andrea-pasquale.

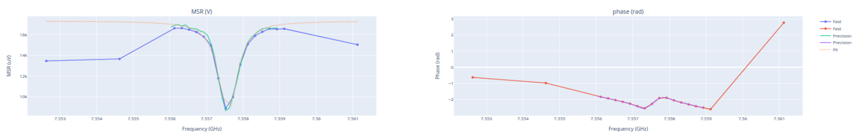
Library	Version
numpy	1.23.4
qibo	0.1.8
qibocal	0.0.1.dev0
qibolab	0.0.1.dev6

Actions

Please find below data generated by actions:

Resonator Spectroscopy

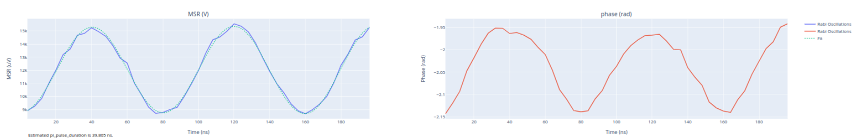
MSR and Phase vs Frequency - Qubit 2



The estimated resonator_freq is 7557585104.9 Hz.
The estimated peak_voltage is 8268.925 uV.

Rabi Pulse Length

MSR vs Time - Qubit 2



Estimated pi_pulse_duration is 39.805 ns.

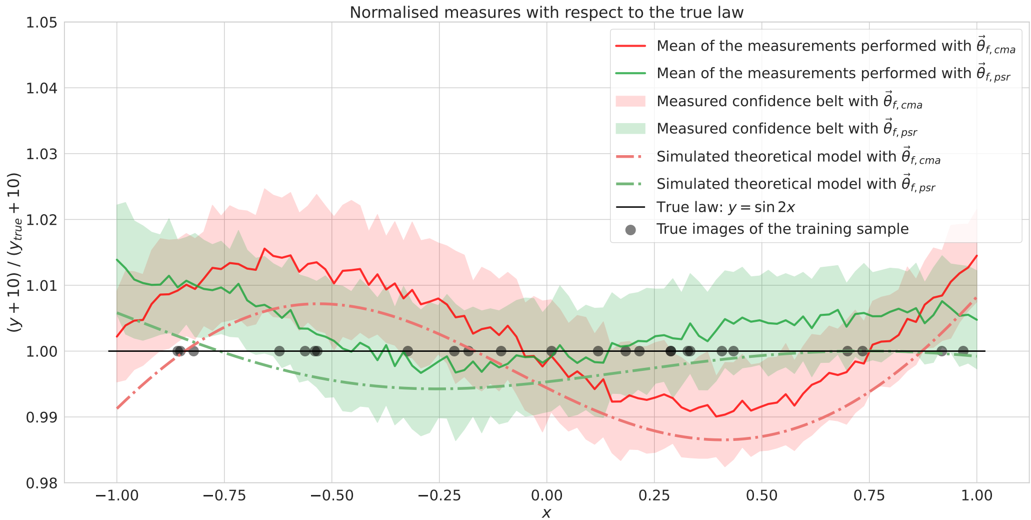
Estimated pi_pulse_max_voltage is 15353.1 uV.

Applications

Applications

What can we achieve using Qibo + Qibolab + Qibocal?

Successfully performed a **gradient descent on a QPU** with one qubit using the Parameter Shift Rule algorithm.



https://arxiv.org/pdf/2210.10787.pdf

Outlook

Qibo is growing to accommodate different tasks:

- ✔ High performance quantum simulation: qibojit
- ✔ Hardware control: qibolab
- ✔ Hardware calibration: **qibocal**

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`

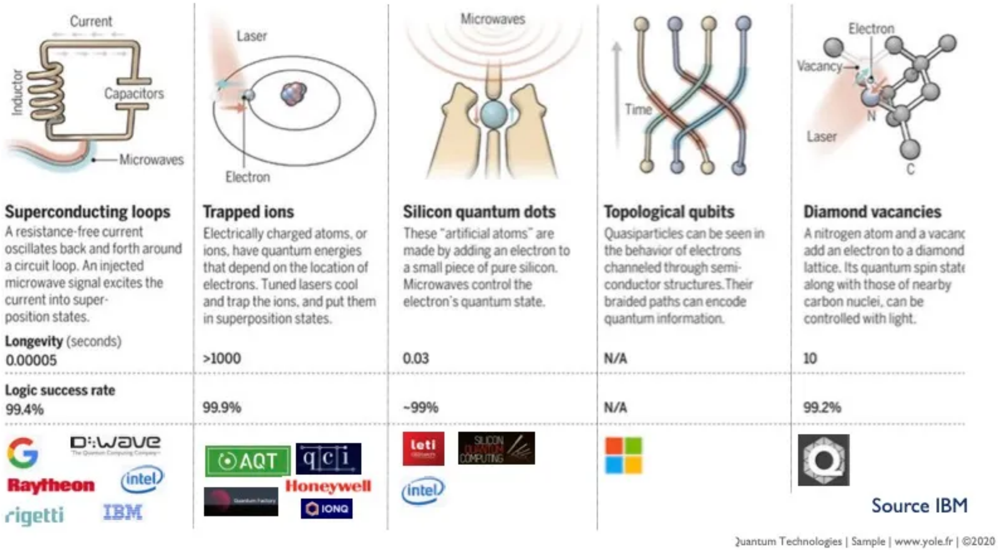
`https://github.com/qiboteam/qibocal`

`https://github.com/qiboteam/qibolab`

Thanks for listening!

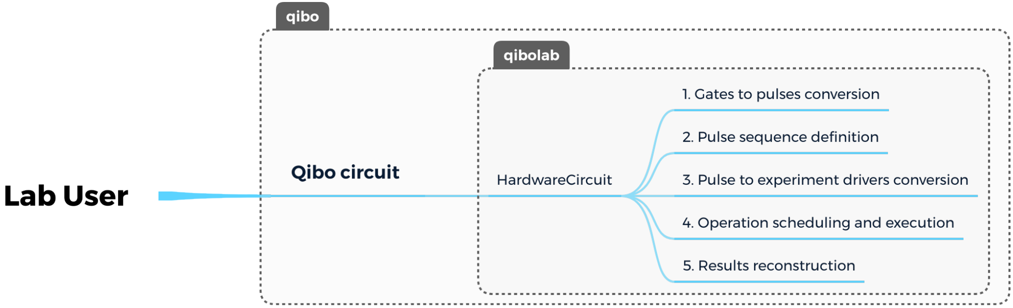
Backup Slides

How can we implement physical qubits?


Current
Inductor
Capacitors
Microwaves
Laser
Electron
Microwaves
Time
Electron
Vacancy
N
Laser
C
Superconducting loops
A resistance-free current oscillates back and forth around a circuit loop. An injected microwave signal excites the current into super-position states.
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.
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.
Topological qubits
Quasiparticles can be seen in the behavior of electrons channeled through semi-conductor structures. Their braided paths can encode quantum information.
Diamond vacancies
A nitrogen atom and a vacanc add an electron to a diamond lattice. Its quantum spin state along with those of nearby carbon nuclei, can be controlled with light.
Longevity (seconds)
0.00005
>1000
0.03
N/A
10
Logic success rate
99.4%
99.9%
~99%
N/A
99.2%
Quantum Technologies | Sample | www.yole.fr | ©2020

Source IBM

How to use qibolab?




```python
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

slurm
workload manager