Towards a hybrid quantum operating system

First Year PhD students workshop

Andrea Pasquale

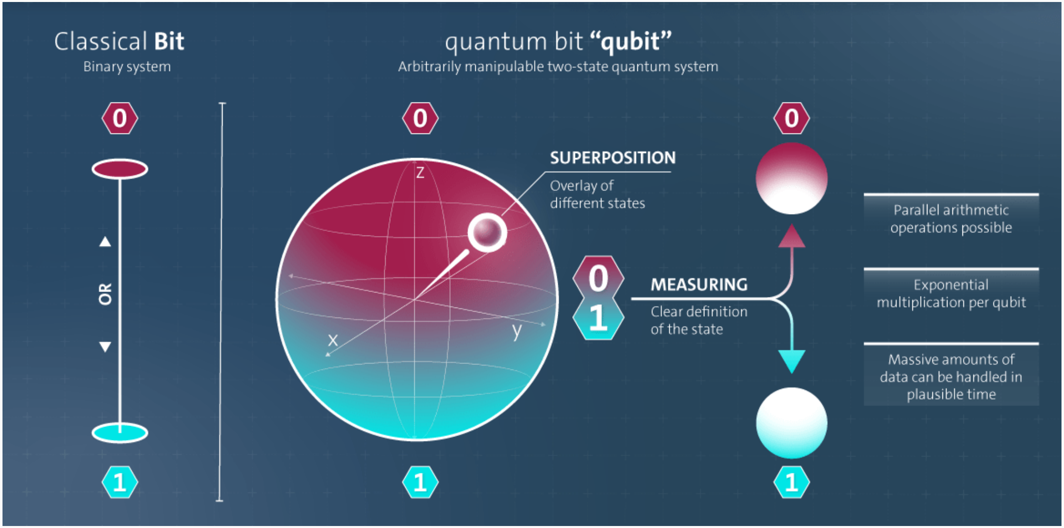
29th September 2022

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

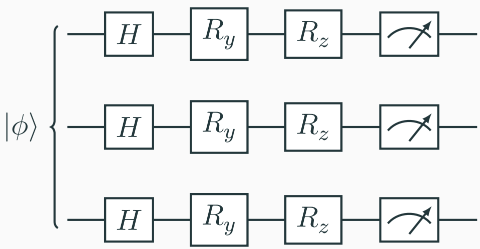
According to Quantum Mechanics (QM) a state ψ evolves through unitary operators U_i

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

In the gates-based model we obtain a circuit:

which we call quantum circuit.

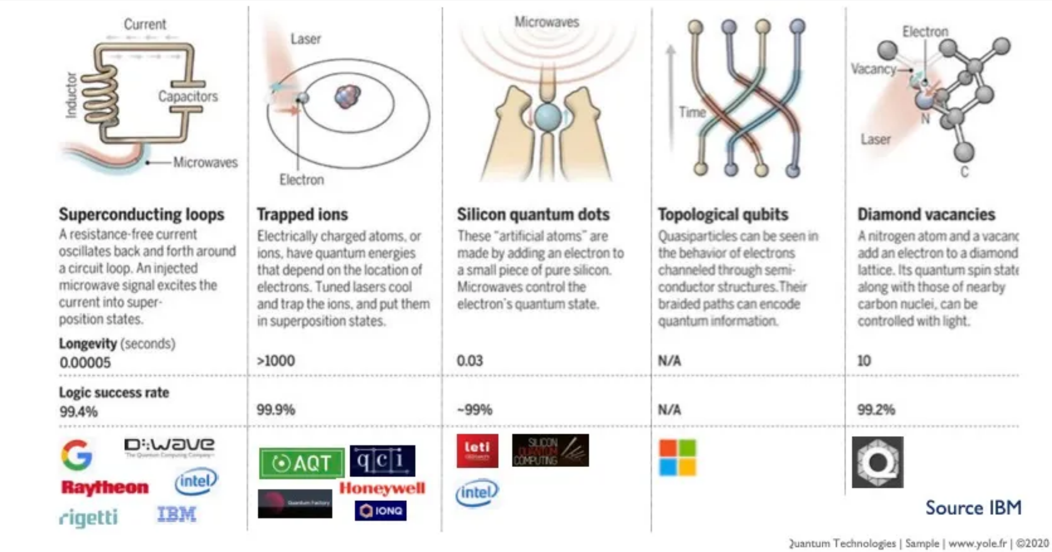
With multiple qubits we get the following:



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

How can we implement physical qubits?

Quantum technologies

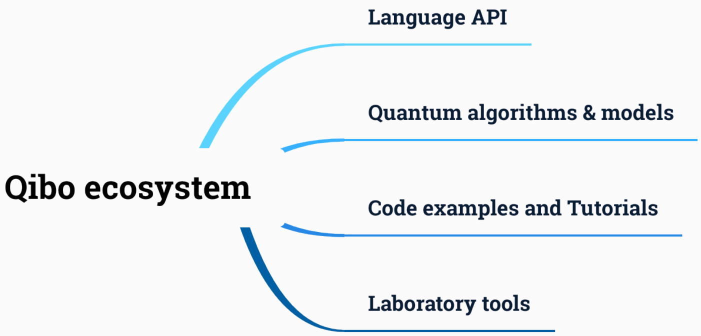


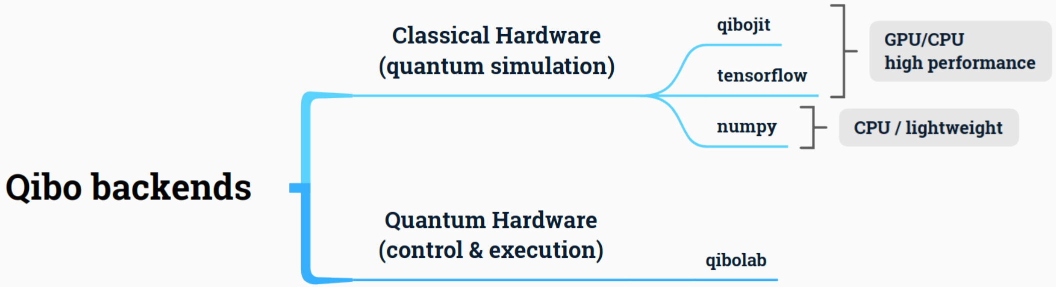
Source IBM

Introducing Qibo

Qibo

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




Qibo backends
Classical Hardware
(quantum simulation)
qibojit
tensorflow
numpy
GPU/CPU
high performance
CPU / lightweight
Quantum Hardware
(control & execution)
qibolab

Introducing Qibojit

Matrix multiplication to simulate circuits:

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

✖ 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/`

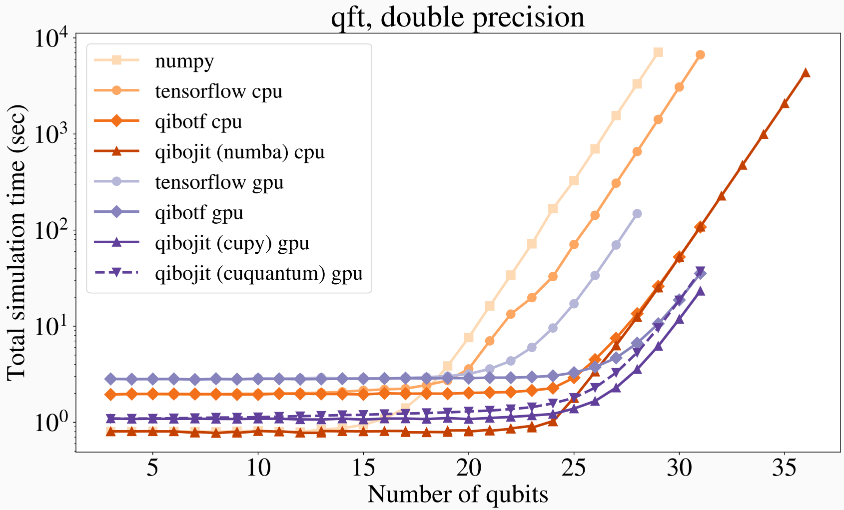
Qibojit - Example

```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
```

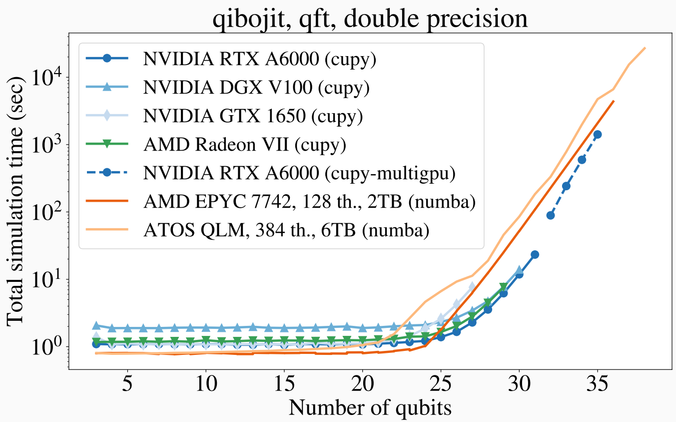
To further speed up:

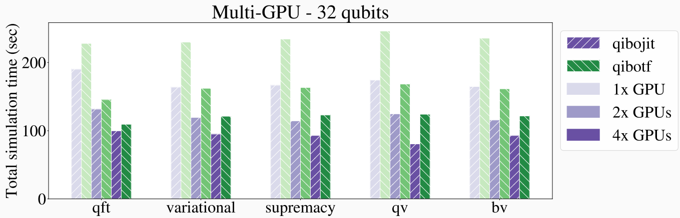
- *in-place updates*
- exploit sparsity of matrices
- specialized operators for:
    - single qubit gate: X, Y Z
    - two qubit gates: SWAP

Benchmarks



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



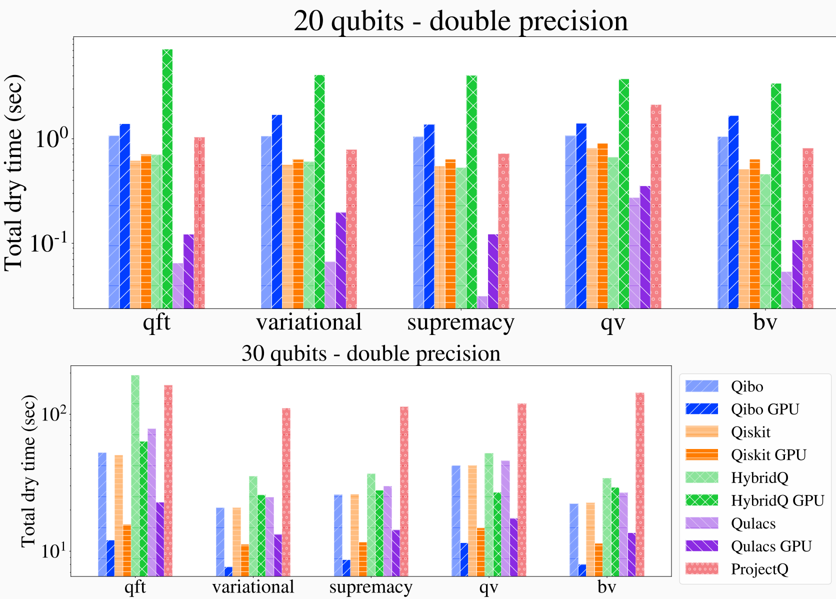


Qibojit features

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

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

How does Qibo perform against the other libraries?

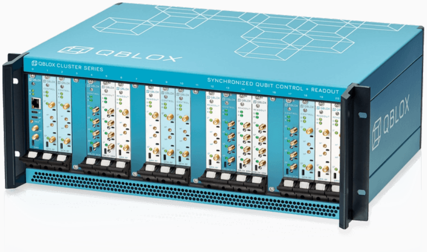


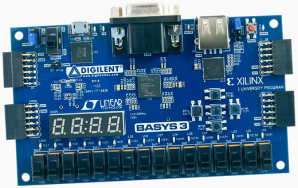
Benchmark library: `https://github.com/qiboteam/qibojit-benchmarks`

Hardware control using Qibo

Hardware control

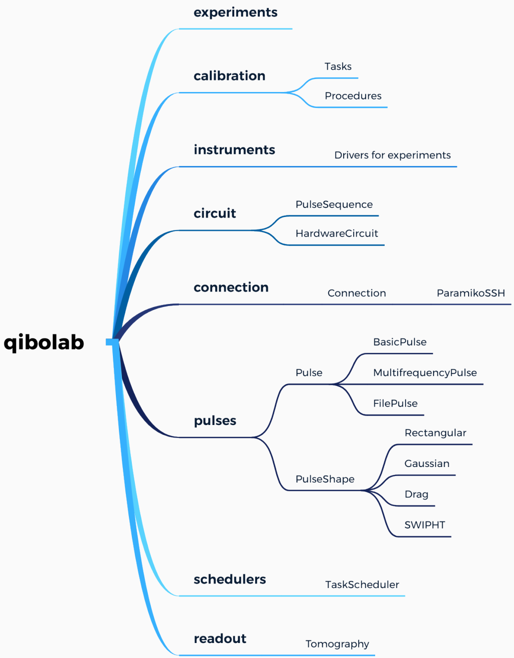
For superconducting qubits **gates** are implemented by sending **pulses**.





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

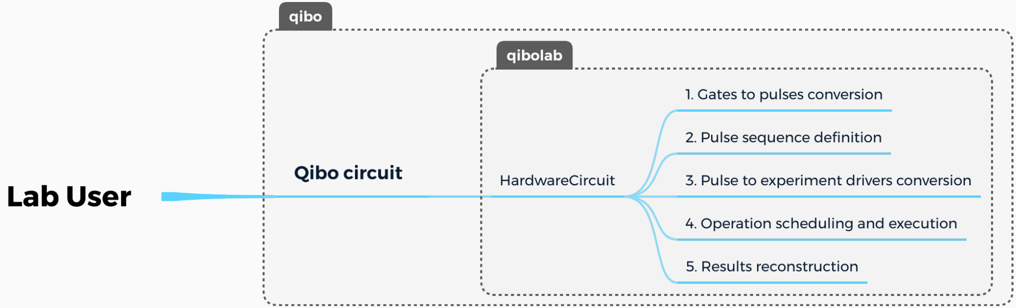
Introducing Qibolab



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?

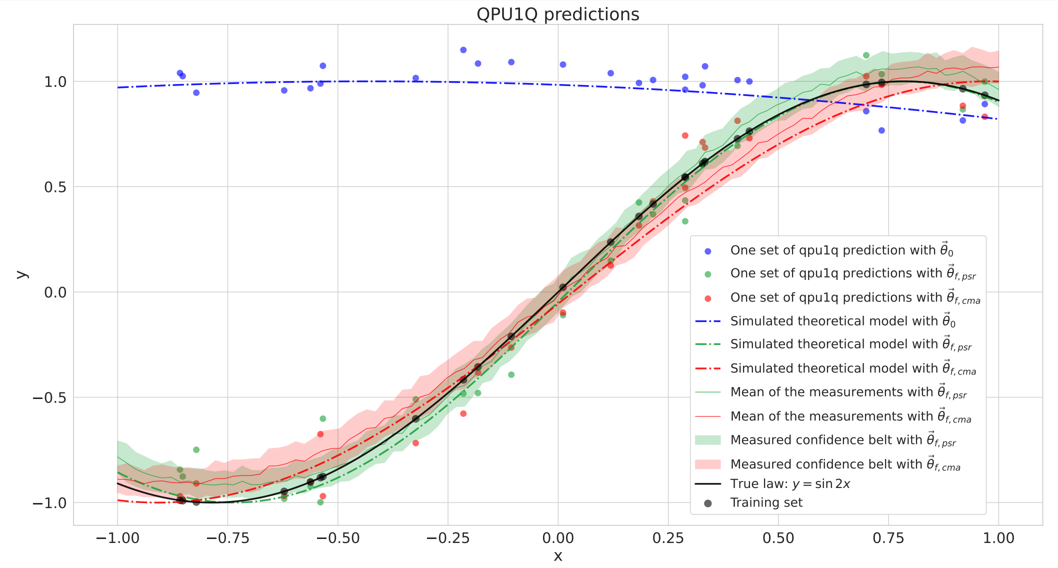



```python
from qibo import models, gates

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

Quantum Machine Learning on Real Hardware

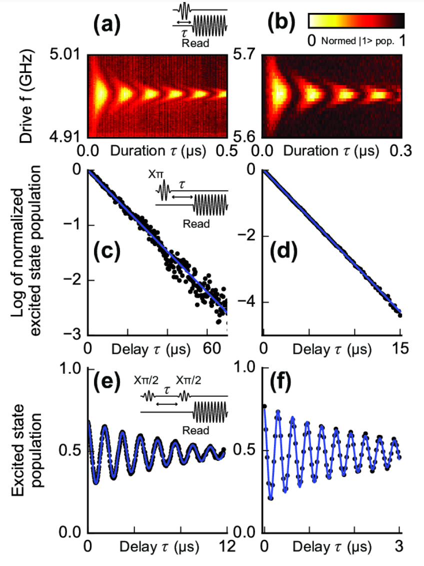


A reporting tool for calibration using Qibo

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):

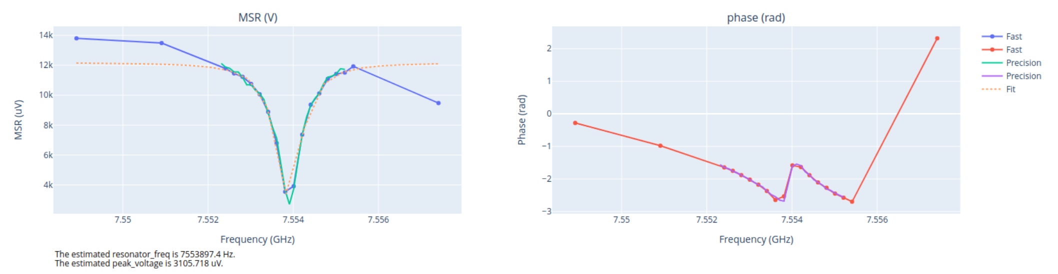
- Perform standard calibration routines:
  - Resonator and qubit spectroscopy
  - Rabi and Ramsey
  - T1 and T2 determination
- Perform quantum protocols to extract the fidelity:
  - Randomized Benchmarking
  - Gate Set Tomography
  - Cross-Entropy Benchmarking
- Repeat the above steps periodically.



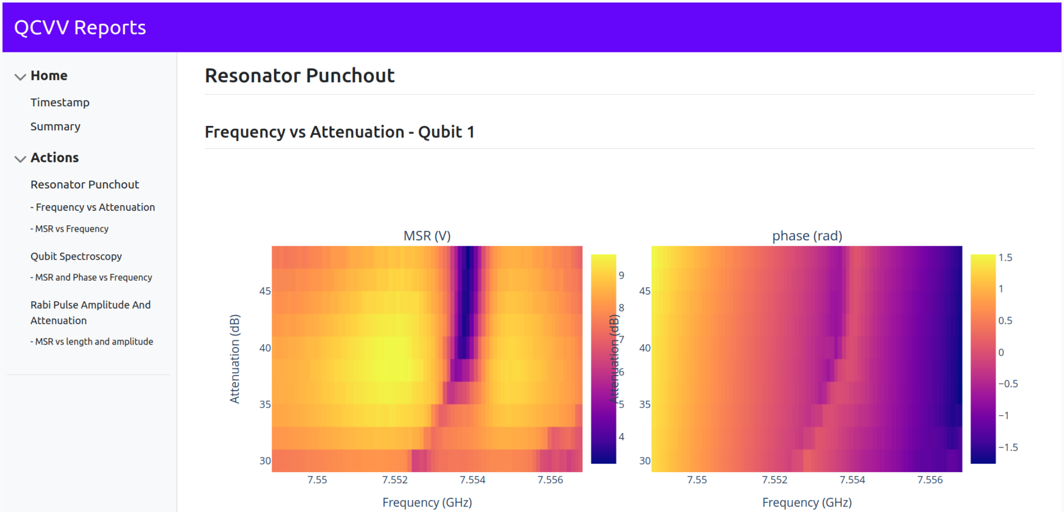
A new reporting tool for Qibo

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



Report example



Outlook

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!