Quantum computing for HEP applications

Challenges and achievements in the NISQ era

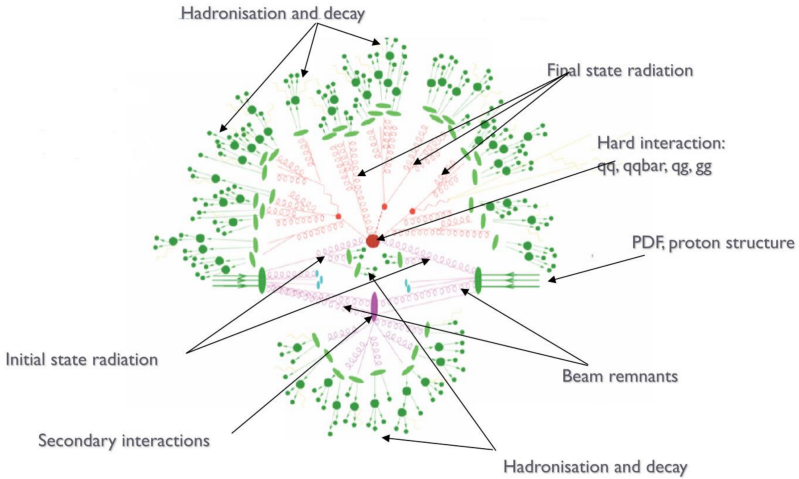
Stefano Carrazza, INFN Milano / Università degli Studi di Milano

September 6th, 2023

Quantum Technologies for Fundamental Physics, EMFCSC, Erice

HEP challenges for LHC and future colliders

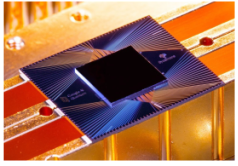
Monte Carlo simulation and data analysis are **intensive** and requires lots of **computing power**.



R&D and adoption of new technologies in HEP

HEP is moving towards new technologies, in particular **hardware accelerators**:

CPU **GPU** **FPGA/ASIC** **Quantum chip**

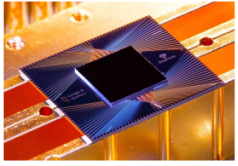


Moving from **general purpose devices** ⇒ **application specific**

R&D and adoption of new technologies in HEP

HEP is moving towards new technologies, in particular **hardware accelerators**:

CPU **GPU** **FPGA/ASIC** **Quantum chip**

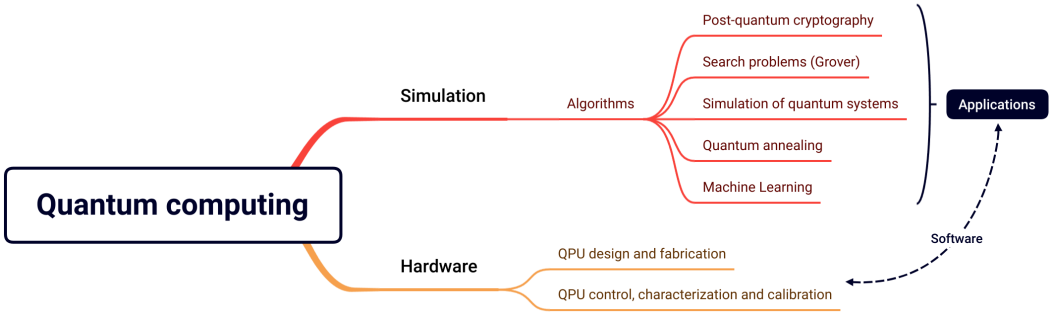


Moving from **general purpose devices** ⇒ **application specific**

Examples of **initiatives** and institutions involved:

CERN | QUANTUM TECHNOLOGY INITIATIVE | INFN | BIQUTE

Quantum Computing topics in HEP



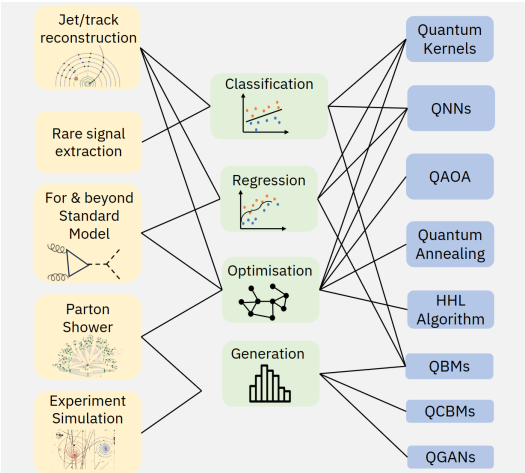
The **HEP community** is testing **quantum computing algorithms** in topics related to:

hep-exp	hep-ph	quant-ph
Data analysis	Theoretical modelling	Software / Middleware

Application examples

Quantum computing for HEP experiments

arXiv:2307.03236



QC4HEP WG

Goal:

Replace **classical ML data analysis** methods with variational quantum computing (QML) and observe **advantage** with quantum computing methods.

How?

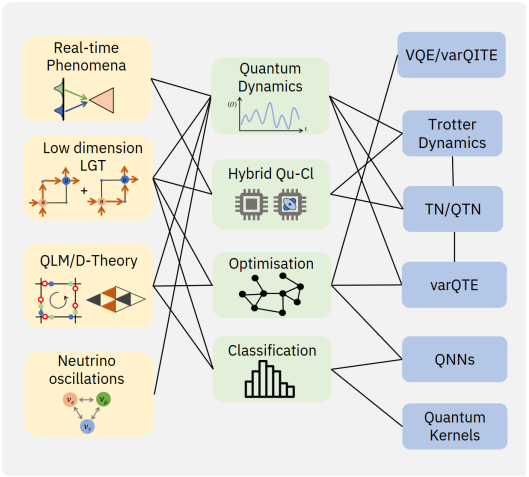
- Developing **variational models** using classical quantum simulation.
- Adapting problems and deploying strategies on **NISQ hardware**.

Goal:

Design **new algorithms** for QFT and Hadronic physics observables, identify **advantage** from quantum computing methods.

How?

- Designing **hybrid quantum-classical** methods using classical quantum simulation.
- Deploying **classical quantum simulation** techniques on HPC infrastructure.



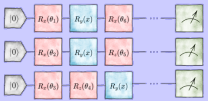
QC4HEP WG

Quantum Machine Learning - ML with Quantum Circuits (credits Robbiati)

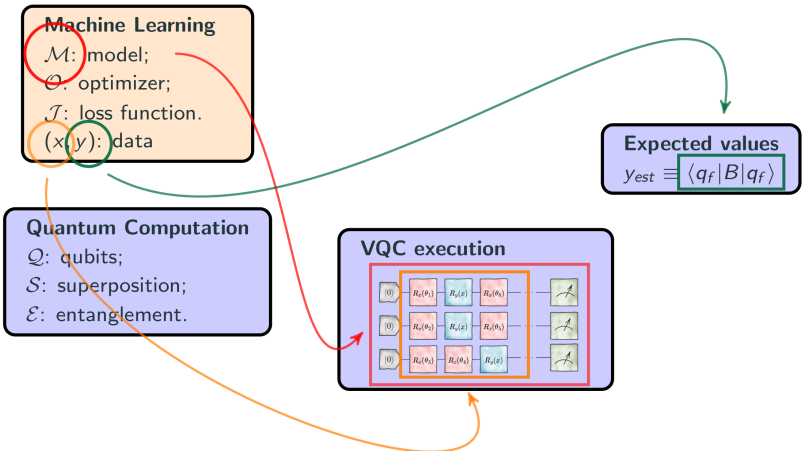
Machine Learning
$\mathcal{M}$: model;
$\mathcal{O}$: optimizer;
$\mathcal{J}$: loss function.
(x, y): data

Quantum Computation
$\mathcal{Q}$: qubits;
$\mathcal{S}$: superposition;
$\mathcal{E}$: entanglement.

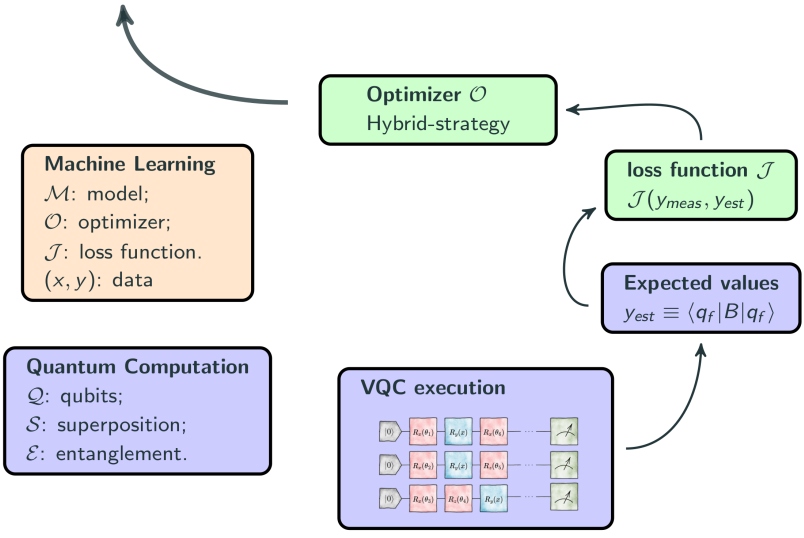
VQC execution



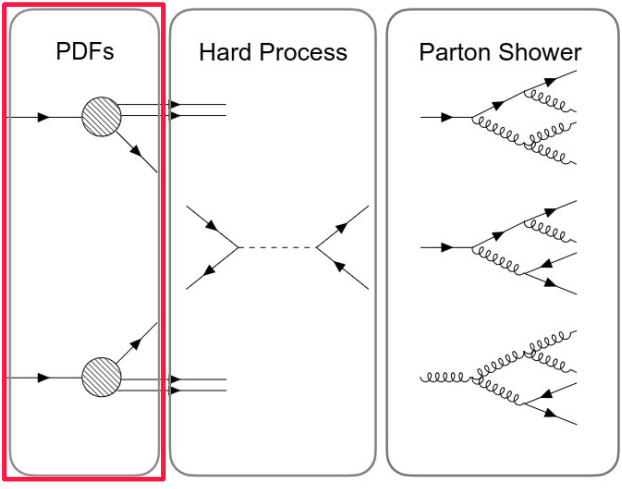
Expected values
$y_{est} \equiv \langle q_f | B | q_f \rangle$


Machine Learning
$\mathcal{M}$: model;
$\mathcal{O}$: optimizer;
$\mathcal{J}$: loss function.
(x, y): data
Expected values
$y_{est} \equiv \langle q_f | B | q_f \rangle$
Quantum Computation
$\mathcal{Q}$: qubits;
$\mathcal{S}$: superposition;
$\mathcal{E}$: entanglement.
VQC execution

Quantum Machine Learning - ML with Quantum Circuits (credits Robbiati)



Example 1: Parton Distribution Functions



Parton distribution functions (Machine Learning)

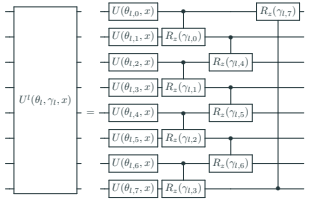
Determination of parton distribution functions

arXiv:2011.13934

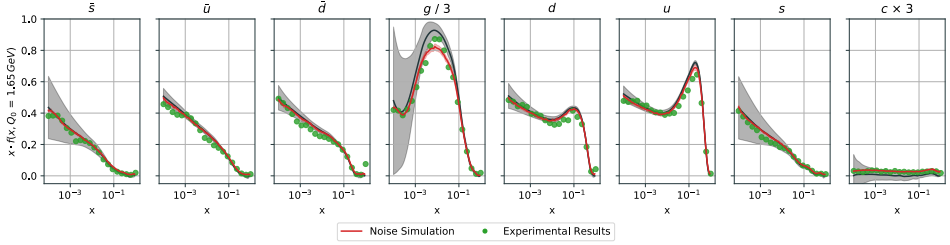
We parametrize **Parton Distribution Functions** with multi-qubit variational quantum circuits:

1. Define a quantum circuit: $\mathcal{U}(\theta, x)|0\rangle^{\otimes n} = |\psi(\theta, x)\rangle$
2. $U_w(\alpha, x) = R_z(\alpha_3 \log(x) + \alpha_4) R_y(\alpha_1 \log(x) + \alpha_2)$
3. Using $z_i(\theta, x) = \langle\psi(\theta, x)|Z_i|\psi(\theta, x)\rangle$:

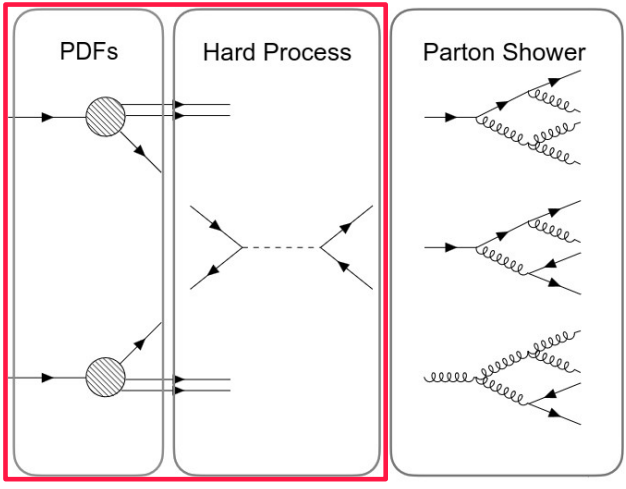
$$\mathrm{qPDF}_i(x, Q_0, \theta) = \frac{1 - z_i(\theta, x)}{1 + z_i(\theta, x)}.$$



Results from **classical quantum simulation and hardware execution** (IBM) are **promising**:



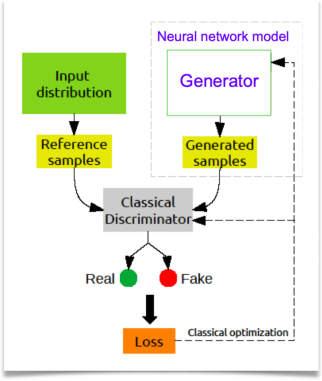
Example 2: Event generation



Event generation

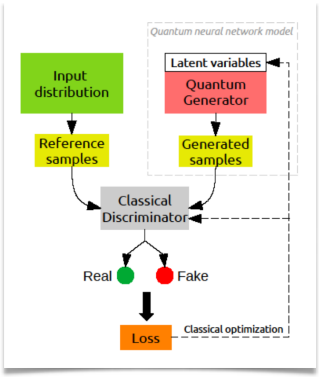
Train with a **small dataset**, use **unsupervised machine learning models** to learn the underlying distribution and generate for free a much larger dataset.

Classical setup:

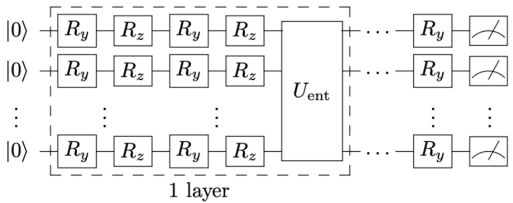


Only the generator becomes quantum

Hybrid quantum-classical setup:



Quantum generator: a series of quantum layers with rotation and entanglement gates



1 component = 1 qubit

$$\boldsymbol{x} = \left(-\left\langle \sigma_z^1 \right\rangle, -\left\langle \sigma_z^2 \right\rangle, \ldots, -\left\langle \sigma_z^n \right\rangle\right)$$

Style-based approach

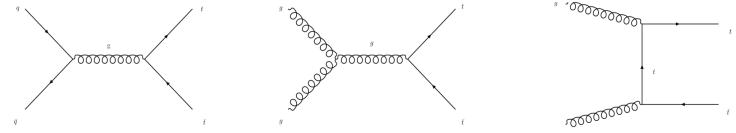
the noise is inserted in **every gate and not only in the initial quantum state**

$$R_{y,z}^{l,m}\left(\boldsymbol{\phi_g}, \boldsymbol{z}\right) = R_{y,z}\left(\phi_g^{(l)} z^{(m)} + \phi_g^{(l-1)}\right)$$

Reminiscent of the reuploading scheme

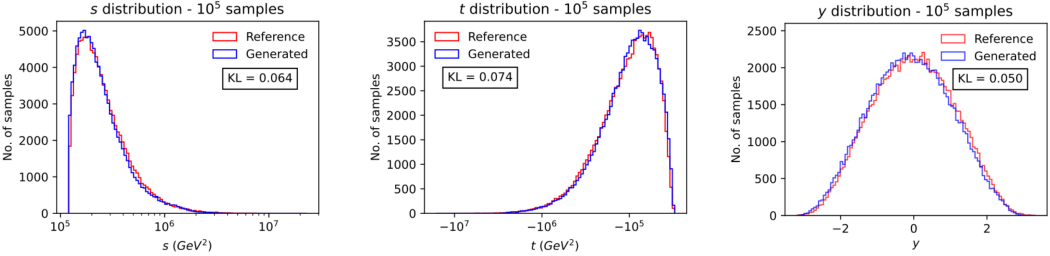
A. Pérez-Salinas, et al., *Quantum* **4**, 226 (2020)

Testing the style-qGAN with real data: proton-proton collision $pp \to t\bar{t}$



Training and reference samples for **Mandelstam variables** (s, t) and rapidity y generated with MadGraph5_aMC@NLO.

Simulation results: 3 qubits, 2 layers, 100 bins



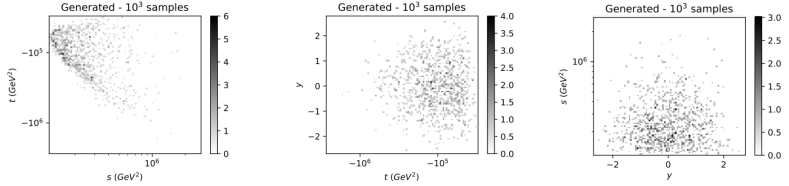
Testing different architectures

arXiv:2110.06933

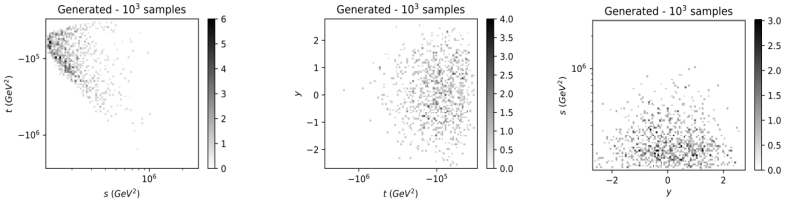
- Access constraints to *ionQ*: test limited to 1000 samples only

Very similar results:
implementation largely hardware-independent

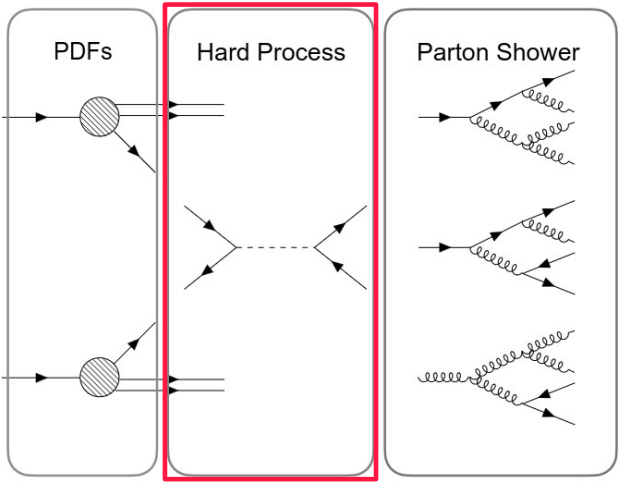
ionQ samples:



IBM Q samples:



Example 3: Monte Carlo Integration / Sampling



Monte Carlo Integration

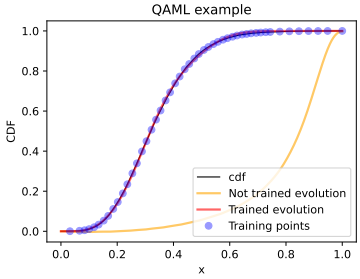
Density estimation with adiabatic QML

arXiv:2303.11346

Determining Probability Density Functions (PDF) by fitting the corresponding Cumulative Density Function (CDF) using an adiabatic QML ansatz.

Algorithm's summary:

1. Optimize the parameters $\boldsymbol{\theta}$ using adiabatic evolution: $H_{\mathrm{ad}}(\tau;\boldsymbol{\theta}) = [1 - s(\tau;\boldsymbol{\theta})]\hat{X} + s(\tau;\boldsymbol{\theta})\hat{Z}$ in order to approximate some target CDF values;
2. Derivate from H_{ad} a circuit $\mathcal{C}(\tau;\boldsymbol{\theta})$ whose action on the ground state of $\hat{X}$ returns $|\psi(\tau)\rangle$;
3. The circuit at step 2 can be used to calculate the CDF;
4. Compute the PDF by derivating $\mathcal{C}$ with respect to τ using the Parameter Shift Rule.

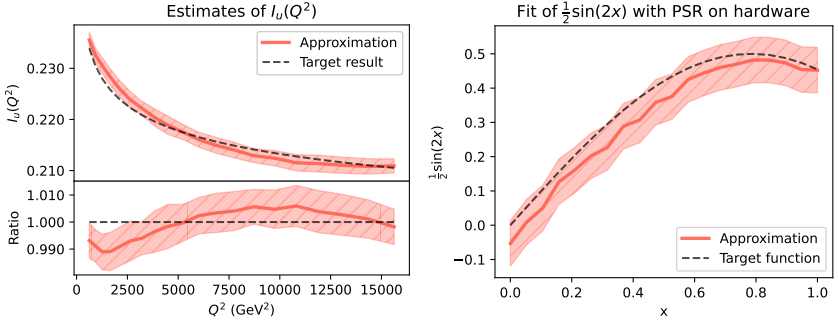


Multi-variable integration with VQCs

arXiv:2308.05657

- Use Variational Quantum Circuits to calculate multi-dimensional integrals: $I(\alpha) = \int_{\boldsymbol{x}_a}^{\boldsymbol{x}_b} g(\alpha; \boldsymbol{x}) \mathrm{d}^n \boldsymbol{x}$.
- Algorithm's summary:

1. Train the derivative of a VQC with respect to the integral variables $\boldsymbol{x}$ to approximate the integrand $g(\boldsymbol{x})$;
2. The derivatives are computed using the Parameter Shift Rule and this allows the same circuit $\mathcal{C}$ to be used for approximating any integrand marginalisation and the primitive.



Middleware challenges

Software challenges

How to **design** quantum algorithms and **deploy** on quantum hardware?

- Cloud-based quantum hardware:
  - Use tools provided by providers.
- Self-hosted quantum hardware:
  - Operate self-hosted quantum hardware.
  - Accommodate custom lab setups.
  - Bypass restrictions to execute an experiment.

Software challenges

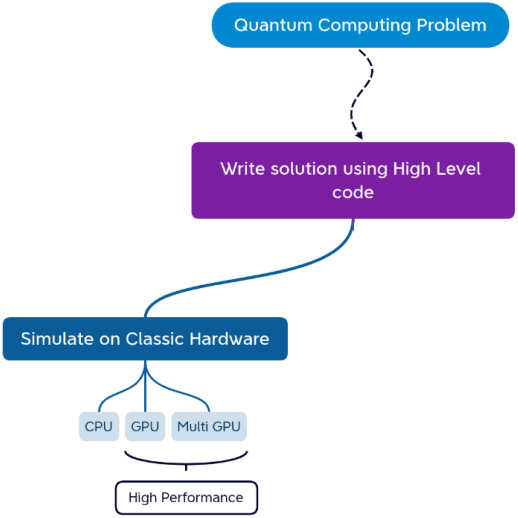
How to **design** quantum algorithms and **deploy** on quantum hardware?

- Cloud-based quantum hardware:
  - Use tools provided by providers.
- Self-hosted quantum hardware:
  - Operate self-hosted quantum hardware.
  - Accommodate custom lab setups.
  - Bypass restrictions to execute an experiment.

Example

Let's consider the PDF determination project, it requires two stages: **prototyping** and **deployment**.

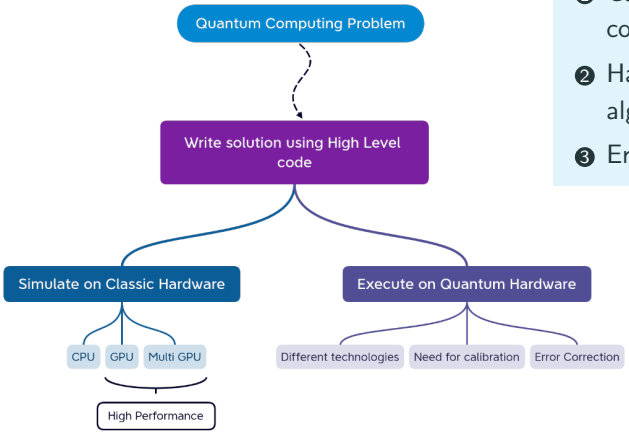
Stage 1: Prototyping models / algorithms



Prototyping

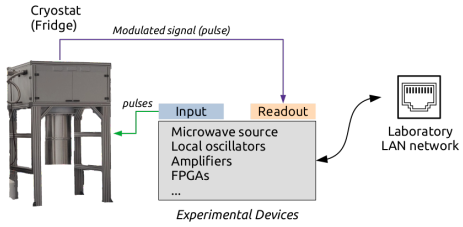
1. High-level quantum circuit programming language
2. Fast classical quantum simulation for model prototyping

Stage 2: Deployment on quantum hardware

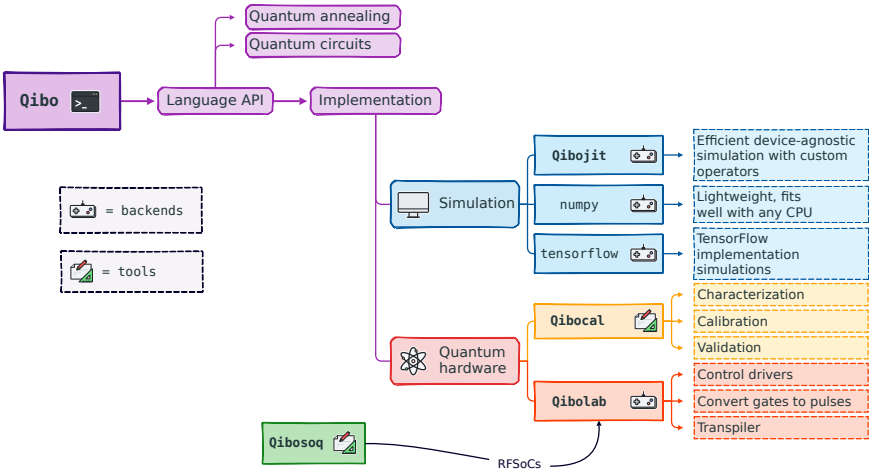


Deployment

1. Gates to microwave pulses sequence compilation (SC qubits)
2. Hardware compatible optimization algorithms
3. Error-mitigation algorithms

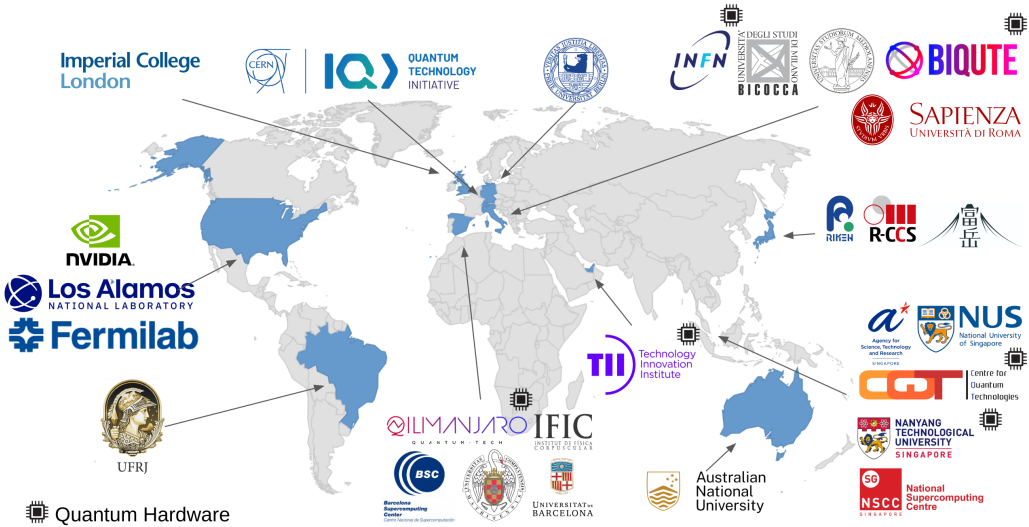


Qibo is an open-source hybrid operating system for self-hosted quantum computers.



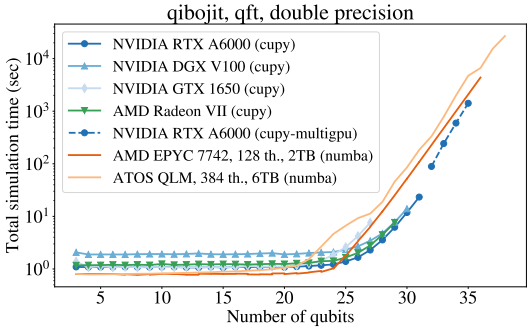
https://qibo.science

Qibo Contributors (September 2023)



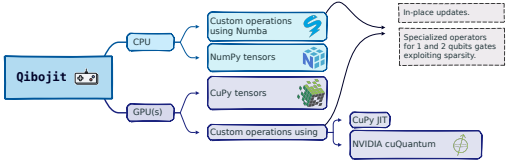
Classical quantum simulation benchmarks

arXiv:2203.08826



Major features:

- Supports CPU, GPU and multi-GPU.
- NVIDIA and AMD GPUs support.
- Reduced memory footprint.

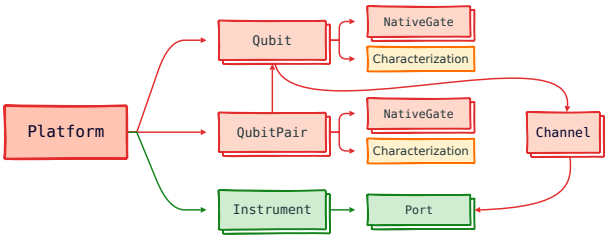


Benchmark library: `https://github.com/qiboteam/qibojit-benchmarks`

How to control qubits?

arXiv:2308.06313

Qibolab is the dedicated Qibo backend for quantum hardware control.

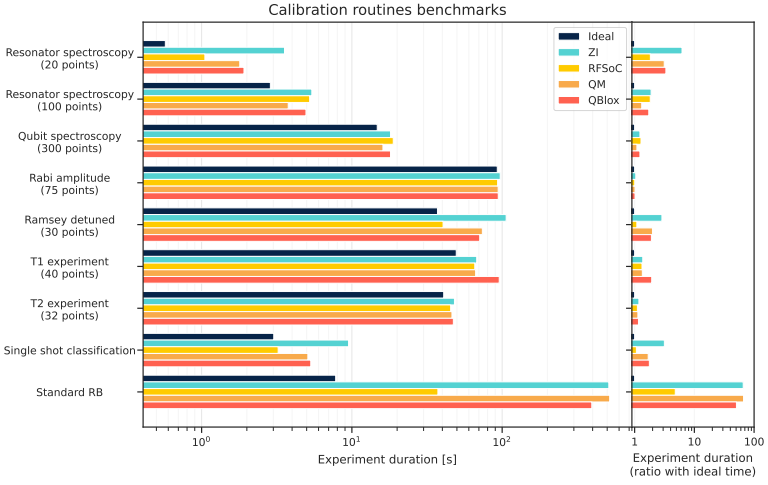


- Platform API: support custom allocation of quantum hardware platforms / lab setup.
- Drivers: supports commercial and open-source firmware for hardware control.
- Arbitrary pulse API: provide a library of custom pulses for execution through instruments.
- Transpiler: compiles quantum circuits into pulse sequences matching chip topology.
- Quantum Circuit Deployment: seamlessly deploys quantum circuit models on quantum hardware.

Benchmarking instruments performance

arXiv:2308.06313

We compare the ideal pulse sequence execution performance to instruments execution duration.

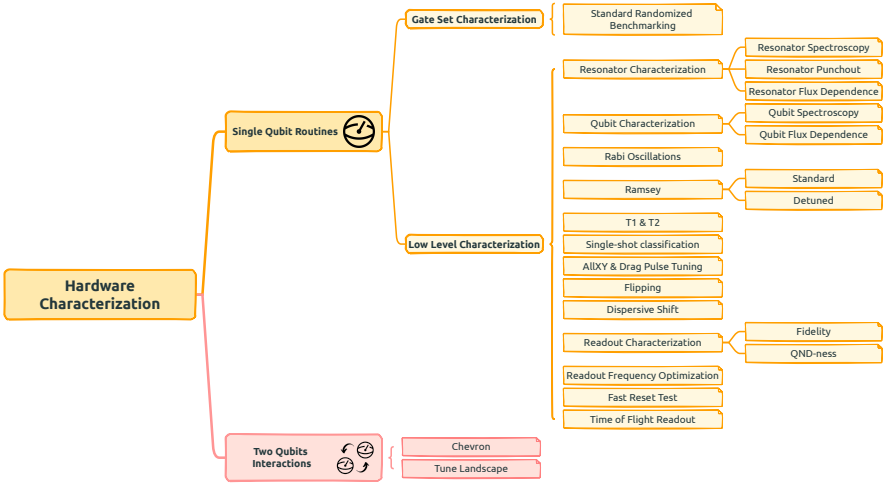


Zurich Instruments (ZI), Quantum Machines (QM), QBlox and RFSoC FPGA (Qibosoq+QICK).

Qubit characterization and calibration

arXiv:2308.06313

Calibration of single and multi-qubit platforms are possible using **Qibo**.



Full-stack procedure: PDF determination

arXiv:2308.06313

1. High-Level API (Qibo)

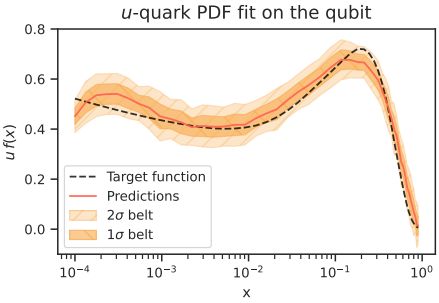
- Define model prototype.
- Implement training loop.
- Perform training using simulation.

2. Calibration (Qibocal)

- Calibrate qubit.
- Generate platform configuration.

3. Execution (Qibolab)

- Allocate calibrated platform.
- Compile and transpile circuit.
- Execute model and return results.



Parameter	Value
N_{data}	50 points
N_{shots}	500
MSE	10^{-3}
Electronics	Xilinx ZCU216
Training time	¡2h

Real-time error mitigation in QML trainings

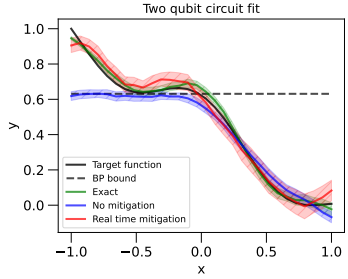
Cleaning up the parameters space with a real-time error mitigation strategy in order to overcome **Noise-Induced Barren Plateaus** (NIBP) when training a QML model:

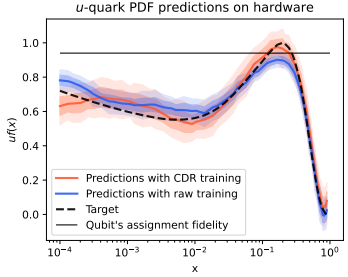
Algorithm's summary:

1. The expected values E are mitigated through **Clifford Data Regression** (CDR):

$$E_{\text{mit}} = \alpha_{\text{cdr}} E_{\text{noisy}} + \beta_{\text{cdr}};$$

2. Reduced CDR computational cost by updating $(\alpha, \beta)_{\text{cdr}}$ periodically during the training;
3. The mitigation **removes the bounds and accelerate the training process**.

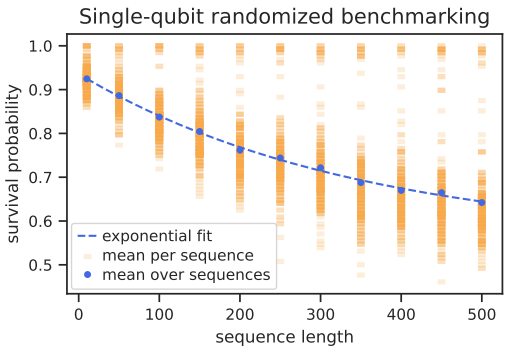


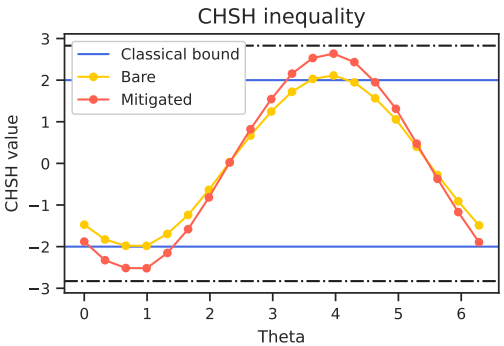


Full-stack deployment: RB and Bell inequality

arXiv:2308.06313

Examples of results executed on quantum hardware (single-qubit) after calibration with Qibo:





Outlook

Outlook

We have observed a great set of interesting **proof-of-concept** applications in HEP.

For the future:

- Improve results quality, moving from **prototype to production**.
- Mitigate hardware noise by implementing **real-time error mitigation** techniques.
- Provide **software tools** for further enhancement of quantum technologies.
- Enhance **calibration, characterization and validation** techniques.
- Codevelop **quantum hardware and instruments** for application specific tasks.