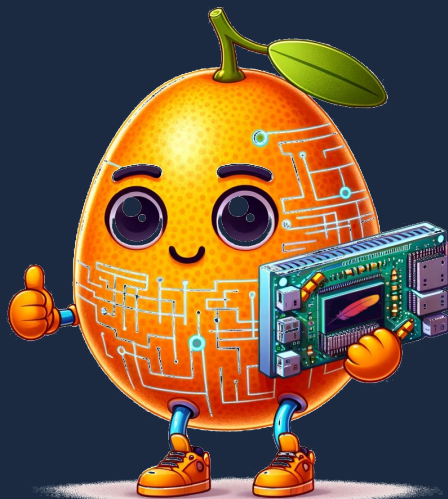


Apache Mahout

Quantum computing in Python with Qumat and QDP

Community Over Code Hackathon

Andrew K Musselman, Krishna Dave



What we will do today



Meet Mahout

An Apache project for quantum computing in Python



Build circuits with Qumat

One API, three simulators: Qiskit, Cirq, Braket



Pick a hackathon task

Twelve one-to-two-hour tasks, no GPU needed



Install in one minute

pip, optional GPU extra, or from source with uv



Encode data with QDP

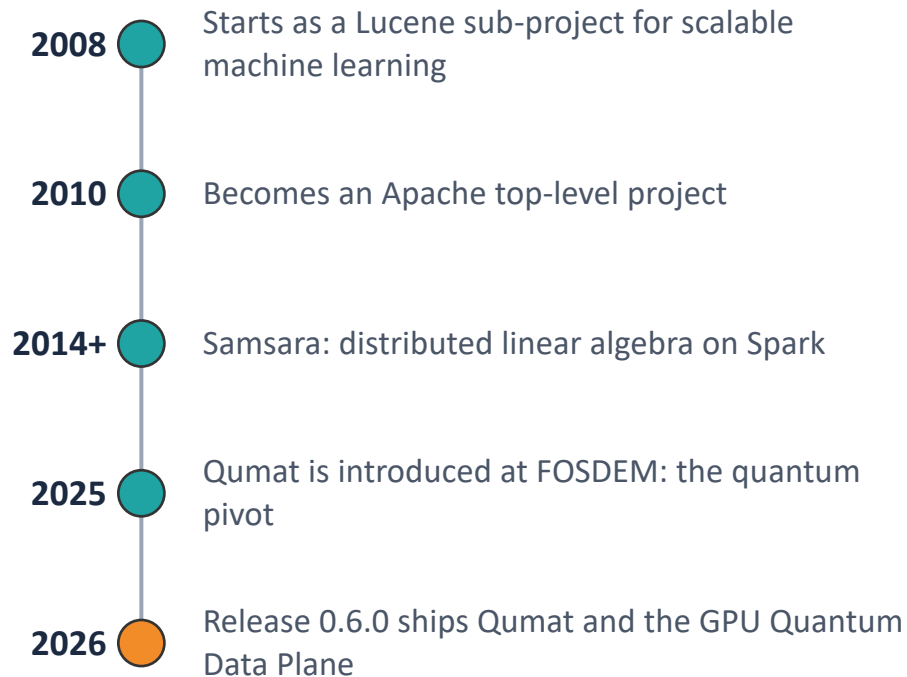
GPU kernels, zero-copy hand-off to PyTorch



Ship a pull request

The Mahout workflow from fork to merge

Apache Mahout, then and now



0.6.0

latest release, May 2026

3

circuit backends: Qiskit,
Cirq, Braket

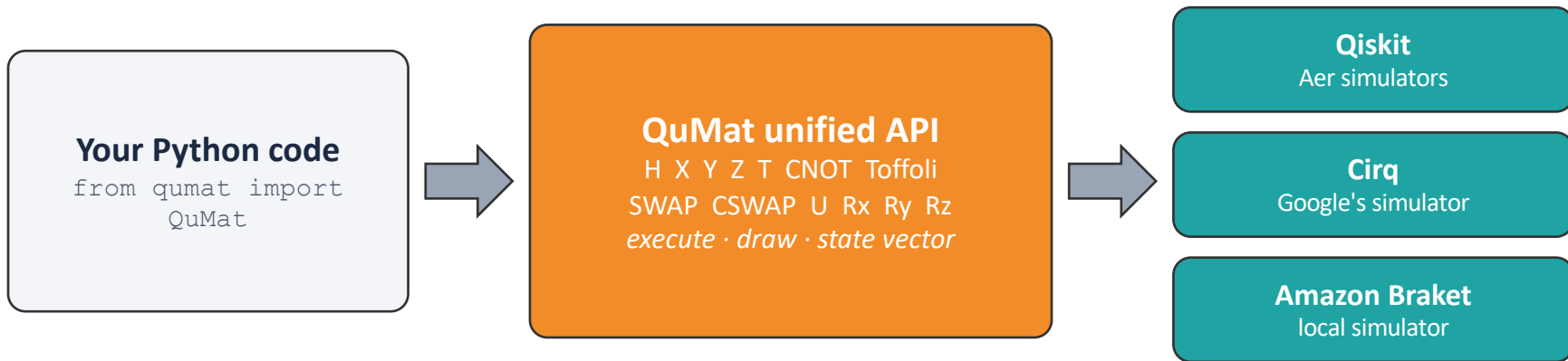
6

GPU encoding methods in
QDP

3.10–3.12

supported Python
versions

Qumat: write once, run on any backend



Same gates everywhere

Switch simulators by changing one config dict



Parameterized circuits

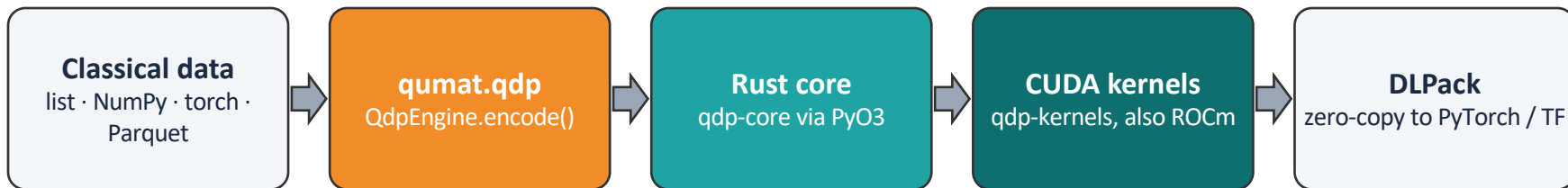
Name angles like "theta", bind values at run time



Built-in swap test

measure_overlap() returns state fidelity

QDP: GPU-accelerated data encoding



Output: a GPU-resident state vector of shape $[batch, 2^n]$, complex64 or complex128, never copied through the host.

Method	What goes in	Typical use
amplitude	up to 2^n values per sample, L2-normalized	feature vectors for QML
angle	one rotation angle per qubit	small dense features
basis	one integer index, $0 \leq i < 2^n$	categorical / lookup states
phase	one phase angle per qubit	phase-based kernels
iqp, iqp-z	n (+ pairwise) parameters	IQP feature maps

Install in one minute

```
# Qumat core, with Qiskit, Cirq and Braket
pip install qumat

# Add QDP: Linux + NVIDIA CUDA or AMD ROCm GPU
pip install "qumat[qdp]"

# From source, for contributors
git clone https://github.com/apache/mahout.git
cd mahout && pip install uv
uv sync                # core
uv sync --extra qdp    # with QDP (needs a GPU)

# Smoke test
python -c "import qumat; print('ok')"
```



Python 3.10 to 3.12

Pure Python wheel for Qumat,
prebuilt Linux wheels for QDP



No GPU needed for Qumat

All three simulators run on a laptop
CPU



GPU for QDP

Any CUDA GPU from Turing on; a free
Colab T4 works



Colab notebook

Today's demo runs in the browser,
nothing to install

Hello, Bell state

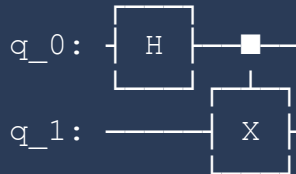
```
from qumat import QuMat

qc = QuMat({
    "backend_name": "qiskit",
    "backend_options": {
        "simulator_type": "aer_simulator",
        "shots": 1024,
    },
})

qc.create_empty_circuit(num_qubits=2)
qc.apply_hadamard_gate(0)
qc.apply_cnot_gate(0, 1)

print(qc.execute_circuit())
{'00': 517, '11': 507}
```

```
print(qc.draw())
```



What happened

H puts qubit 0 in superposition, CNOT entangles qubit 1 with it. Measuring 1024 times gives only 00 and 11, each about half the time.

Counts are a plain dict, so the rest of your Python stack just works.

Same circuit, three backends

```
def bell(backend_name, simulator_type):
    qc = QuMat({
        "backend_name": backend_name,
        "backend_options": {
            "simulator_type": simulator_type,
            "shots": 100,
        },
    })
    qc.create_empty_circuit(2)
    qc.apply_hadamard_gate(0)
    qc.apply_cnot_gate(0, 1)
    return qc.execute_circuit()

bell("qiskit", "aer_simulator")
bell("cirq", "default")
bell("amazon_braket", "local")
```

Backend	Result
qiskit	{'00': 52, '11': 48}
cirq	[Counter({0: 51, 3: 49})]
amazon_braket	Counter({'00': 50, '11': 50})

Same gates, same counts, one line changed.

Cirq reports integer basis states (3 = 11) while Qiskit and Braket return bit strings, so keep that in mind when you post-process.

Hackathon idea: a helper that normalizes results across backends.

Parameterized circuits for optimization

```
import numpy as np

qc = QuMat(qiskit_config)
qc.create_empty_circuit(1)
qc.apply_rx_gate(0, "theta")    # symbolic parameter
qc.apply_ry_gate(0, "phi")

for theta in np.linspace(0, np.pi, 5):
    params = {"theta": theta, "phi": 0.3}
    counts = qc.execute_circuit(params)
    p1 = counts.get("1", 0) / 1024
    print(f"theta={theta:.2f}    P(1)={p1:.2f}")

theta=0.00    P(1)=0.02
theta=1.57    P(1)=0.49
theta=3.14    P(1)=0.98
```

1

Name the angle

Pass a string instead of a number to any Rx, Ry, Rz or U gate

2

Bind at run time

execute_circuit(parameter_values=...) or bind_parameters()

3

Optimize

Wrap it in a cost function and let SciPy or PyTorch drive the loop

This is the building block for VQE, QAOA and quantum ML models.

QDP in four lines

```
import torch
from qumat.qdp import QdpEngine

engine = QdpEngine(device_id=0)           # CUDA by default
qtensor = engine.encode(
    [0.5, 0.5, 0.5, 0.5],
    num_qubits=2,
    encoding_method="amplitude",
)
state = torch.from_dlpack(qtensor)        # zero-copy, on GPU
print(state.shape, state.dtype, state.device)
torch.Size([1, 4]) torch.complex64 cuda:0

# Batches and files work too
engine.encode(batch.cuda(), num_qubits=8,
              encoding_method="angle")
engine.encode("data.parquet", num_qubits=10,
              encoding_method="amplitude")
```



Kernels do the math

Normalization and state writes run as CUDA kernels



Feed it anything

Lists, NumPy, torch tensors, Parquet, Arrow, .numpy, .pt



Train right after

The DLPack tensor plugs straight into a PyTorch model

Pick a task: one to two hours each, no GPU

Track A: Qumat, Python only

A1 Clear error for an unknown backend name
ModuleNotFoundError leaks today

A2 Unsupported simulator_type should raise
both Cirq and Braket just print()

A3 Cirq backend ignores its initialized simulator
hard-coded cirq.Simulator()

A4 Shared parameter name fails on Qiskit only
name conflict adding parameter 't'

A5 create_empty_circuit() without num_qubits
Qiskit raises, others allocate

A6 Add one missing gate: S, S†, T†, CZ or Phase
one gate per person, three backends

Track B: QDP

B1 #1469 Validate QdpBenchmark arguments
pure Python

B2 #1331 Skip GPU tests when no CUDA device
pure Python, test hygiene

B3 #1468 Honor the loader batch limit
one-line wrapper fix plus a test

B4 #1472 Read LZ4-compressed Feather files
Rust reader, needs cargo installed

B5 #787 NumPy file size check against OOM
Rust reader, needs cargo installed

B6 #1436 VecDeque in StreamingProducer
Rust, needs cargo installed

Claim a task by commenting on its issue. Rust tasks: start cargo build right away (5 to 10 min)

From idea to merged pull request



Fork and branch

from apache/mahout on
GitHub



uv sync

pre-commit install



Test

uv run pytest



Open the PR

link the issue, fill the
template



Review and merge

one committer approval
required



Website and docs

mahout.apache.org



Mailing list

dev@mahout.apache.org



Code and issues

github.com/apache/mahout



Quantum primer

mahout.apache.org/docs/learning

Let's build

Install, run the notebook, claim an issue, open a PR.

github.com/apache/mahout

dev@mahout.apache.org

