Large-scale Hybrid Quantum Workflows with PennyLane

Lee J. O’Riordan (lee@xanadu.ai)
Senior Quantum Software Dev. \ PennyLane Performance Lead
Pennylane is the leading open-source library for researchers to build state-of-the-art hybrid & device agnostic quantum algorithms.

GitHub:PennyLaneAI/pennylane
We work with a lot of people, in a lot of places...

Hardware
Fabrication partners and component suppliers.

Software
Channel and software partners for Xanadu Cloud and PennyLane.

Applications
Quantum application and algorithm development.
…to support Quantum programming on any platform

- Integrated Photonics
- Trapped Ions
- ML Tools
- Superconducting Qubits
- Quantum Simulators
- Platforms

- Xanadu
- Quantum Nanodiamond
- Quantinuum
- AQT
- IONQ
- PyTorch
- Numpy
- TensorFlow
- IBM
- Rigetti
- Google
- Microsoft
- AWS
- HPC
...with compositions of hybrid Quantum & Classical Computations...

Support for arbitrary hybrid quantum-classical models; every PennyLane computation is end-to-end differentiable

Trainable circuit + Trainable classical model = Rich, end-to-end trainable hybrid model
...with help from Quantum Nodes (QNode)

QNodes interface between quantum computers and classical scientific libraries

Native Tensor object
torch.Tensor
tf.Tensor

Unwrap and extract numerical value

QNode

evaluate()
gradient()

Convert back to a Tensor object

NumPy
PyTorch
TensorFlow
JAX
Plenty of examples to go around

State preparation with pytorch

3-qubit Ising model

QAOA

VQE

Modeling chemical reactions

Givens rotations for quantum chemistry

QAOA for MaxCut

Building molecular hamiltonians

Quantum transfer learning

Quantum natural gradient

Barren plateaus in quantum neural networks

Quantum advantage with GBS

Beyond classical computing with qsim

Variational classifier

See more at pennylane.ai/qml
PennyLane on HPC platforms
Focus on best tools for the task at hand

** PENNYLANE **

** DISTRIBUTED **

** INDUSTRY STANDARD FOR HPC TOOLING **
- OpenMP

** ML FRAMEWORKS INTEGRATION **
- PyTorch
- TensorFlow
- JAX

** DISTRIBUTED WORKLOADS **
- Ray
- Dask
# PennyLane-Lightning: run performant simulations on all machines, great (HPC) and small (laptop)

<table>
<thead>
<tr>
<th>Device</th>
<th>Platforms</th>
<th>Unique feature</th>
</tr>
</thead>
</table>
| lightning.qubit | ● Windows: x86_64  
● MacOS: x86_64, ARM  
● Linux: x86_64, ARM, PPC, etc.  

pip install pennylane-lightning | ● Modern C++20 codebase — compiles & runs everywhere a supported compiler does  
● Batching of observable gradients (OpenMP)  
● Automatically dispatched SIMD gate kernels  
● Comes by default with `pip install pennylane`

| lightning.gpu   | ● Linux w/ CUDA: x86_64, ARM*, PPC*  

pip install pennylane-lightning[gpu]  

pip install cuquantum | ● Optimal NVIDIA GPU performance through cuQuantum  
● Native GPU support for adjoint backpropagation  
● Batching of observable gradients over multiple GPUs

| lightning.kokkos| ● Mac & Linux: All CPUs & GPUs supported by Kokkos-Core (Intel, NVIDIA, AMD, etc)  
● PennyLaneAI/pennylane-lightning-kokkos | ● Multithreaded gate applications (OpenMP, C++ threads)  
● Accelerator device execution models natively supported (CUDA, HIP/ROCm, SYCL)  

*Note: ARM and PPC support may vary depending on the specific Linux distributions.*
Goal: Build tooling that is useful

Variational quantum optimization problems (with circuit cutting)
Variational algorithms & gradients
Quantum circuits natively differentiable

Both finite difference and parameter-shift methods have the form:

\[ \Delta f_\theta = f(\theta_1) - f(\theta_2) \]

For N parameters, number of evaluations is \( O(2N) \)
Detour into Circuit cutting: A Tensor Network is a Quantum Circuit is a Tensor Netw...

- Break a large circuit into multiple smaller circuits
- Circuits must be “stitched” back together classically (sums/contractions)
- Multiple smaller circuits can be run independently of one another

Q: Can we farm these circuits out to some combination of CPUs/GPUs/QPUs?

A: Yes! But first, let’s see an example.
Quantum programs can be cut into fragments and executed with very small changes to code

```
import pennylane as qml
from pennylane import numpy as np

# Two qubit device
dev = qml.device("lightning.qubit", wires=2)

@qml.cut_circuit  # Enable circuit cutting
@qml.qnode(dev)

def circuit(param):  # Build 3-qubit circuit
    qml.RX(param, wires=0)
    qml.RY(0.9, wires=1)
    qml.RX(param, wires=2)
    qml.CZ(wires=[0, 1])
    qml.RY(-0.4, wires=0)
    qml.WireCut(wires=1)  # Add wire-cut
    qml.CZ(wires=[1, 2])

    return qml.expval(qml.PauliZ(0) @ qml.PauliZ(1) @ qml.PauliZ(2))
```

- Create a two qubit device
- Enable the circuit cutting functionality
- Build a 3 qubit circuit
- Specify wire-cuts
Circuit execution model

\[ \Theta \rightarrow |0\rangle \rightarrow |f(\Theta)\rangle \]
Circuit execution model

\[ \nabla \theta f = f(\theta_1) - f(\theta_2) \]
Quantum parameter optimization for QAOA

1. Calculate the variational energy for a 129 qubit QAOA clustered graph problem

2. Variational parameter optimization of a 62 qubit QAOA problem
   - For analytical results, see paper arxiv:2207.14734

Fast quantum circuit cutting with randomized measurements

Angus Lowe,1 Matija Medvidović,1,2,3 Anthony Hayes,1 Lee J. O’Riordan,1
Thomas R. Bromley,1 Juan Miguel Arrazola,1 and Nathan Killoran1

1Xanadu, Toronto, ON, M5G 2C8, Canada
2Center for Computational Quantum Physics, Flatiron Institute, 162 5th Avenue, New York 10010, USA
3Department of Physics, Columbia University, New York 10027, USA
Numerical results

1. Variational energy calculation upto 129 qubits

2. Parameter optimization on 62 qubits

2 QAOA circuit layers (p=2)
25 nodes per cluster (n=25)
1 node per inter-cluster connection (k=1)

Clusters varied to increase qubit requirements (r)

(2.a) p=1, n=20, k=1, r=3: ~0.5 hr @ 2 GPU nodes
(2.b) p=2, n=20, k=1, r=3: ~12 hr @ 10 GPU nodes
Scaling results for a 79 qubit problem

p=1, n=25, k=2, r=3 vs GPU resources

GitHub: XanaduAI/randomized-measurements-circuit-cutting
Thank you!

// lee@xanadu.ai

xanadu.ai  //  pennylane.ai

arxiv:2207.14734

GitHub:XanaduAI/randomized-measurements-circuit-cutting