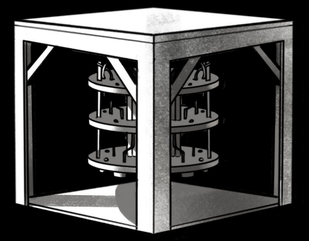
QORO

Scaling quantum computing, with the networks of **today** and **tomorrow**



Monolithic quantum computers don’t scale!

Quantum computers need **millions of qubits**

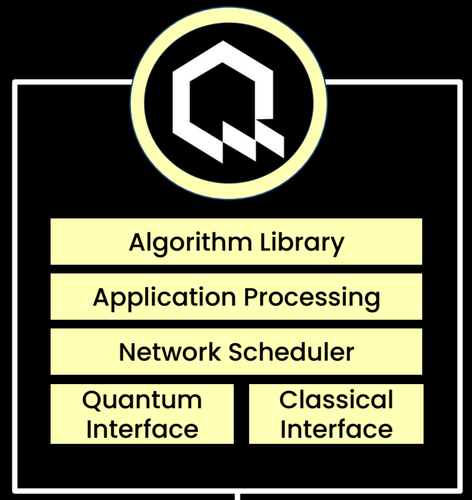
Scaling qubit counts is **fundamentally limited**

Network stacks are required distributed computing

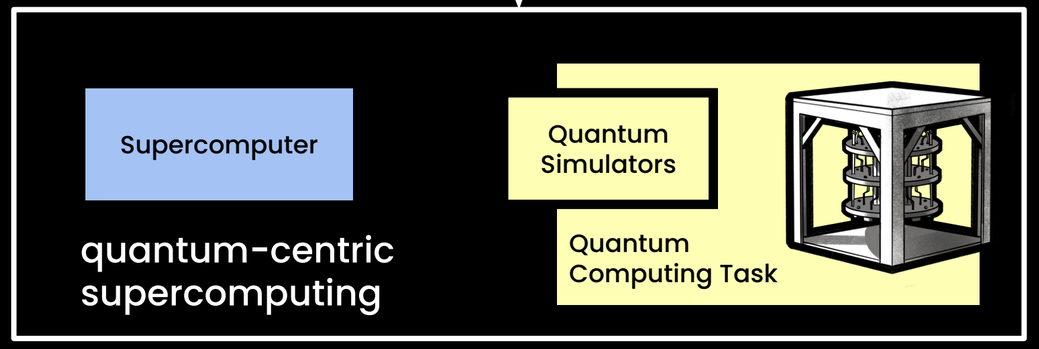
Qoro has *the* solution

today

End-Users

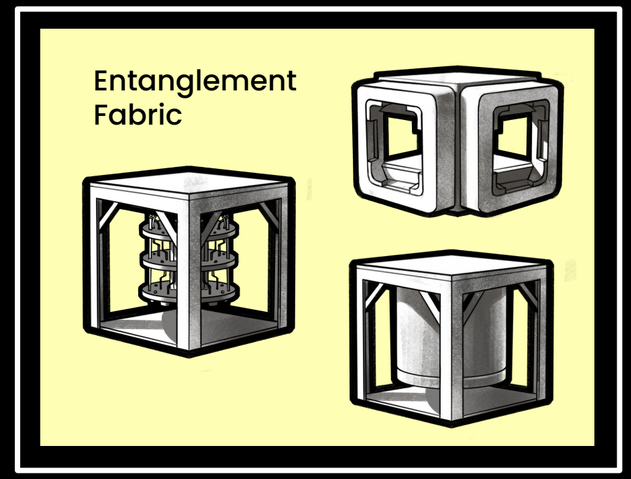


HPC



tomorrow

Quantum Network Operating System



Example: Protein Folding

```
# Define the Hamiltonian as a string
hamiltonian_string = "1613.5895000000003 * IIIIIIII + ... + 5.0 *
ZIIIIIIIZ"
coeffs, observables = convert_hamiltonian(hamiltonian_string)
hamiltonian = qml.Hamiltonian(coeffs, observables)

# Setup the Qoro Service
qoro_service = QoroService(QORO_API_KEY, qpu_system_name="ibm",
shots=15_000, use_circuit_packing=True)

# Define the optimizer
optimizer = MonteCarloOptimizer(n_param_sets=100, n_best_sets=10)

# Build the VQE problem
vqe_problem = VQE(
    hamiltonian=hamiltonian,
    n_electrons=8,
    n_layers=2,
    optimizer=optim,
    ansatz=QAOAAnsatz(),
    max_iterations=15,
    grouping_strategy="qwc",
    backend=qoro_service,
)
vqe_problem.run()
```

What’s happening?

```python
# Define the Hamiltonian as a string
hamiltonian_string = "1613.5895000000003 * IIIIIIIII + ... + 5.0 * ZIIIIIIIZ"
coeffs, observables = convert_hamiltonian(hamiltonian_string)
hamiltonian = qml.Hamiltonian(coeffs, observables)

# Setup the Qoro Service
qoro_service = QoroService(QORO_API_KEY, qpu_system_name="ibm", shots=15_000, use_circuit_packing=True)

# Define the optimizer
optimizer = MonteCarloOptimizer(n_param_sets=100, n_best_sets=10)

# Build the VQE problem
vqe_problem = VQE(
    hamiltonian=hamiltonian,
    n_electrons=8,
    n_layers=2,
    optimizer=optim,
    ansatz=QAOAAnsatz(),
    max_iterations=15,
    grouping_strategy="qwc",
    backend=qoro_service,
)
vqe_problem.run()
```

Divi

Build out the quantum program

Reduce the workload

Send to Qoro Cloud

Composer

Allocate resources

Hardware parallelization

Schedule

Conductor

Interface with hardware

Create and monitor jobs

Post-process hardware results

Application processing

Huge Cost Savings

Costs for running a 17 qubit protein folding problem on IBM (PAYG, $96/min):

Conditions:

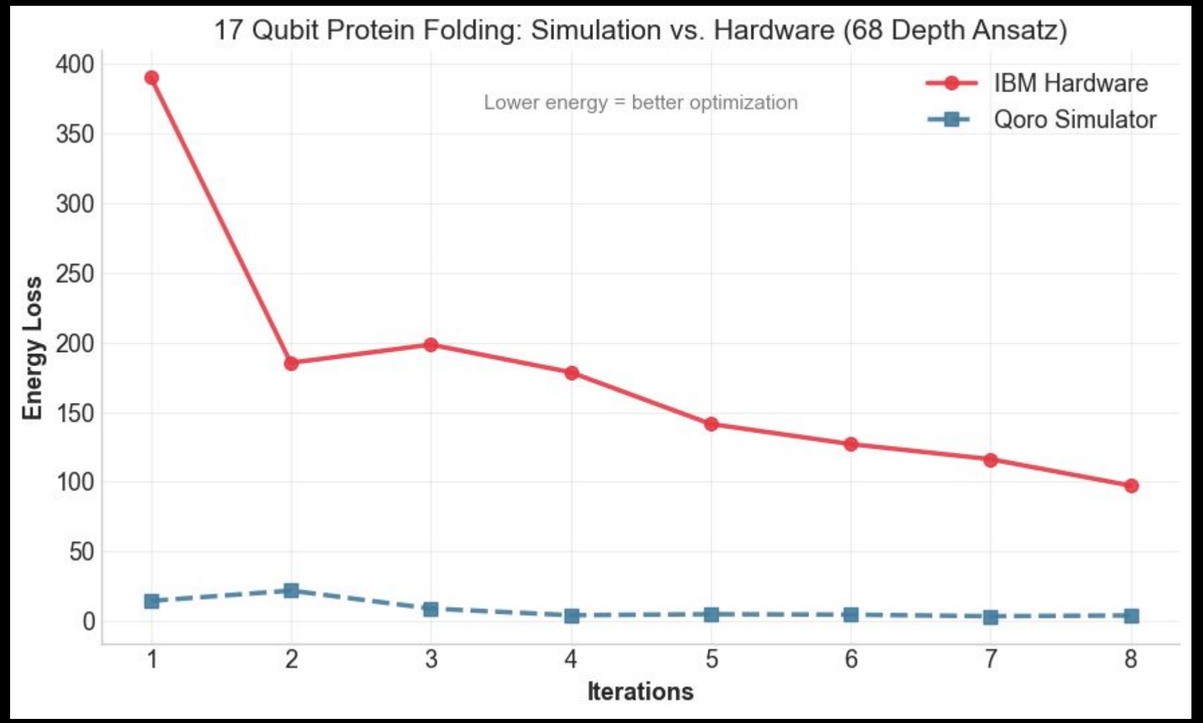
- Use Monte-Carlo optimization (gradient free)
- Using a 335 term Hamiltonian
- Using standard VQE

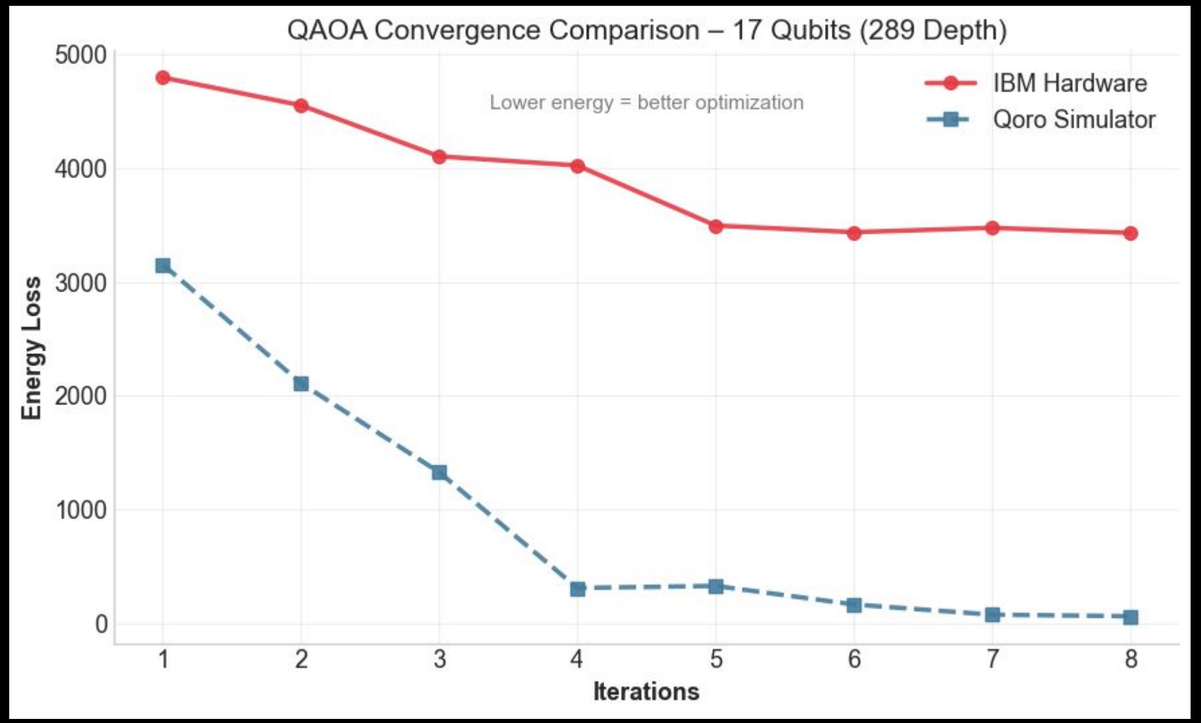
Huge Cost Savings

PAYG, $96/min **PER ITERATION!**

Tool Used	Lines of code	Number of circuits	Cost for 8,000 shots	QPU Runtime	Reduction
Naive implementation	> 2,000	33,500	$128,640	22 hours	
Divi & Grouping	20	100	$384	4 minutes	99.7%
Composer & Packing	0	100	$48	30 seconds	99.9%

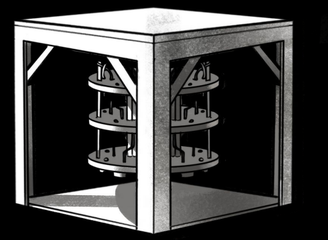
Quality


17 Qubit Protein Folding: Simulation vs. Hardware (68 Depth Ansatz)
Lower energy = better optimization
IBM Hardware
Qoro Simulator
Energy Loss
Iterations
400
350
300
250
200
150
100
50
0
1
2
3
4
5
6
7
8


QAOA Convergence Comparison – 17 Qubits (289 Depth)
Lower energy = better optimization
IBM Hardware
Qoro Simulator
Energy Loss
Iterations
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
7
8

QORO

Scaling quantum computing, with the networks of **today** and **tomorrow**



team@qoroquantum.de