Cloud-Native Quantum: Running Quantum Serverless Workloads on Kubernetes

Paul Schweigert & Michael Maximilien, IBM

Speakers

Paul Schweigert (psschwei.com)

Senior Software Engineer at IBM

Knative Technical Oversight Committee
Kubernetes Contributor

Studied history / played poker

Dr. Max (@maximilien)

Distinguished Engineer at IBM

CTO Open Quantum and Open Serverless

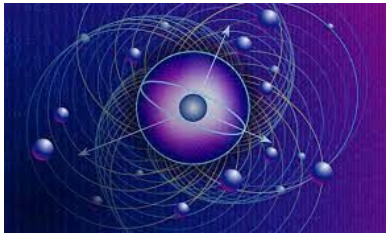
Cyclist / photographer

Agenda

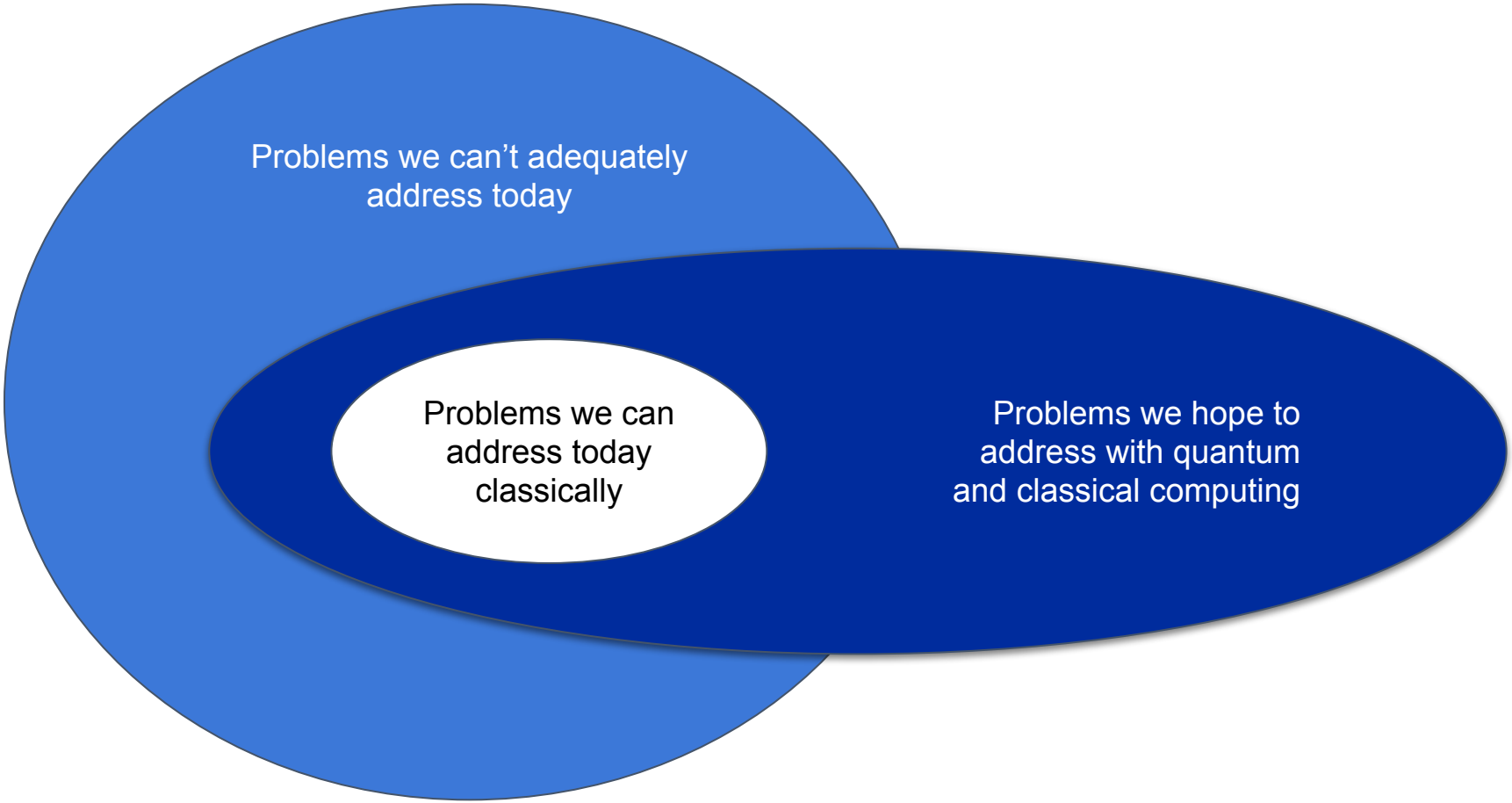
1. What is quantum computing?
2. How do we do quantum computing?
3. How do we do **cloud-native** quantum computing?

$$|\phi\rangle\langle\psi| \doteq \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_N \end{pmatrix} \begin{pmatrix} \psi_1^* & \psi_2^* & \cdots & \psi_N^* \end{pmatrix} = \begin{pmatrix} \phi_1\psi_1^* & \phi_1\psi_2^* & \cdots & \phi_1\psi_N^* \\ \phi_2\psi_1^* & \phi_2\psi_2^* & \cdots & \phi_2\psi_N^* \\ \vdots & \vdots & \ddots & \vdots \\ \phi_N\psi_1^* & \phi_N\psi_2^* & \cdots & \phi_N\psi_N^* \end{pmatrix}$$

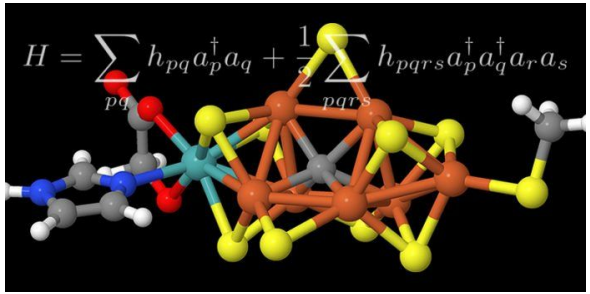
What is quantum computing?

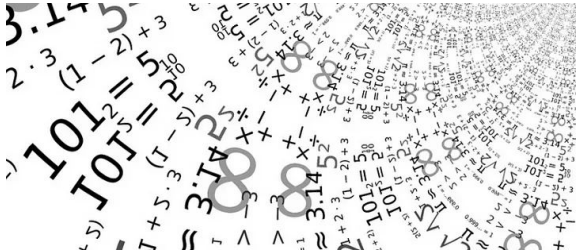


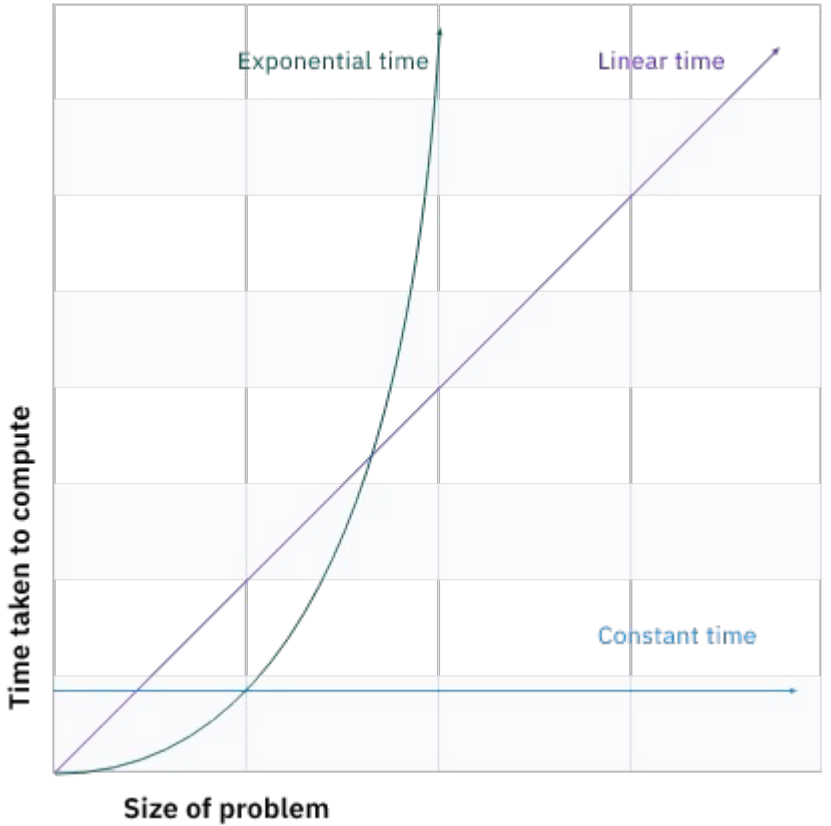
Why quantum?


Problems we can't adequately address today
Problems we can address today classically
Problems we hope to address with quantum and classical computing

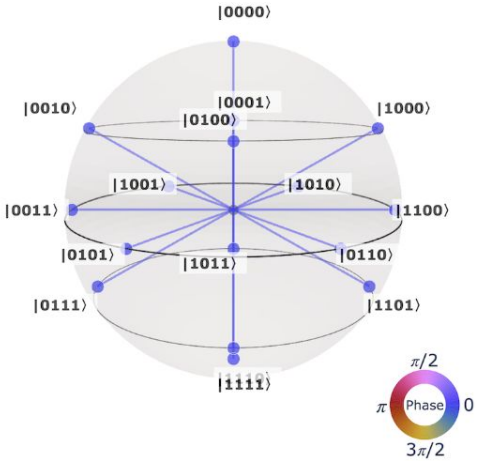
Some examples…



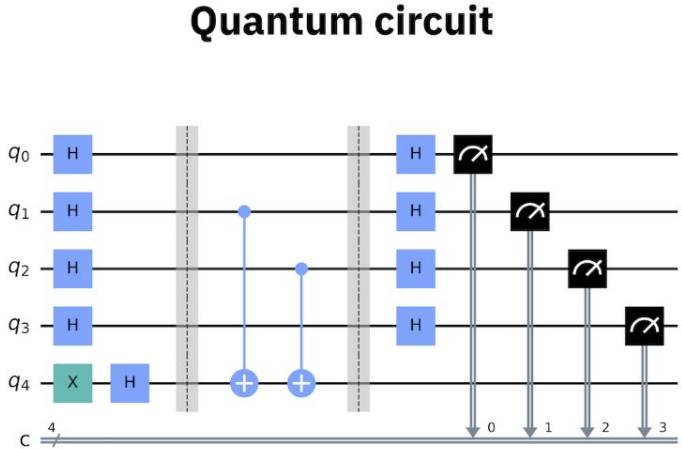




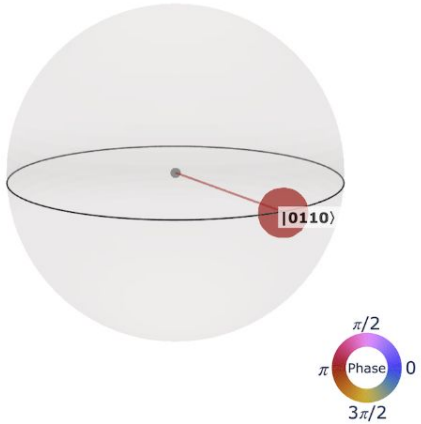
Quantum computers use qubits



Superposition of all possibilities

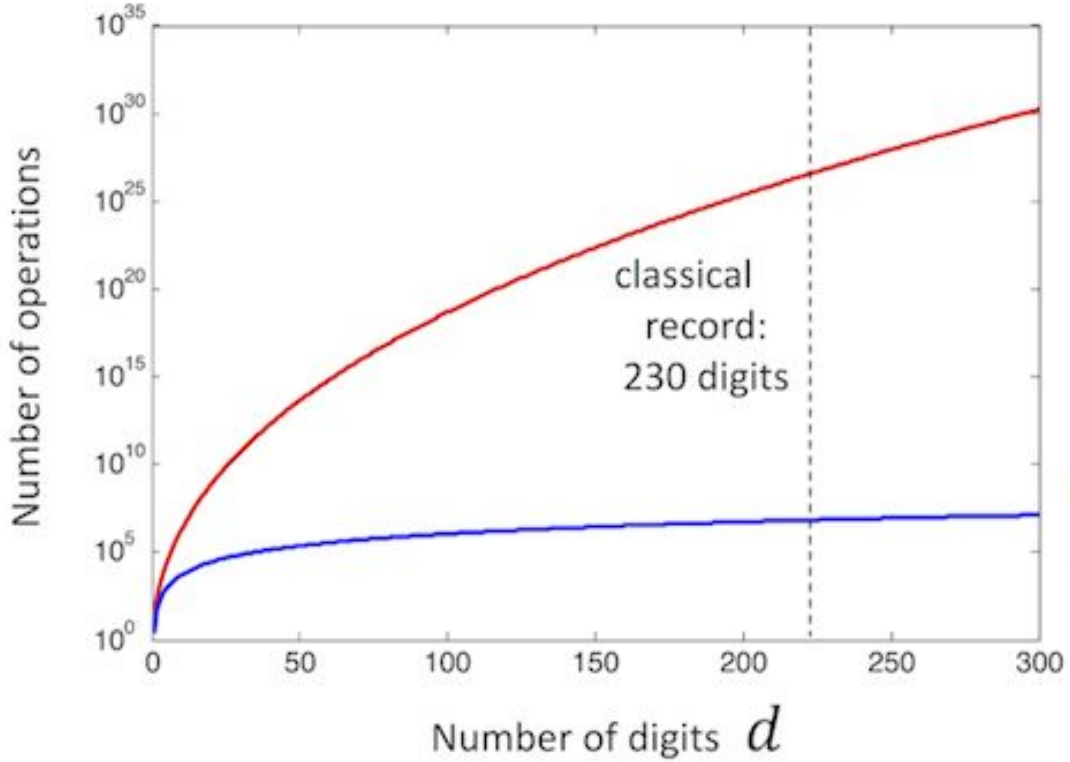


Computation driven interference



Solution

Ex: Shor's algorithm for factoring

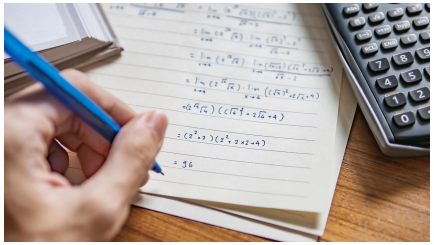


$\exp(\text{const} \times d^{1/3})$

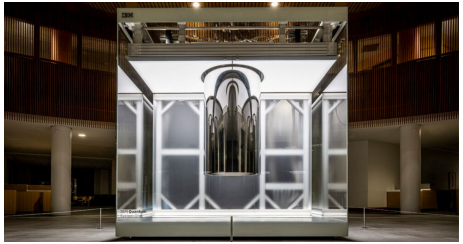
best classical
algorithm
(number field sieve)

$\text{const} \times d^3$

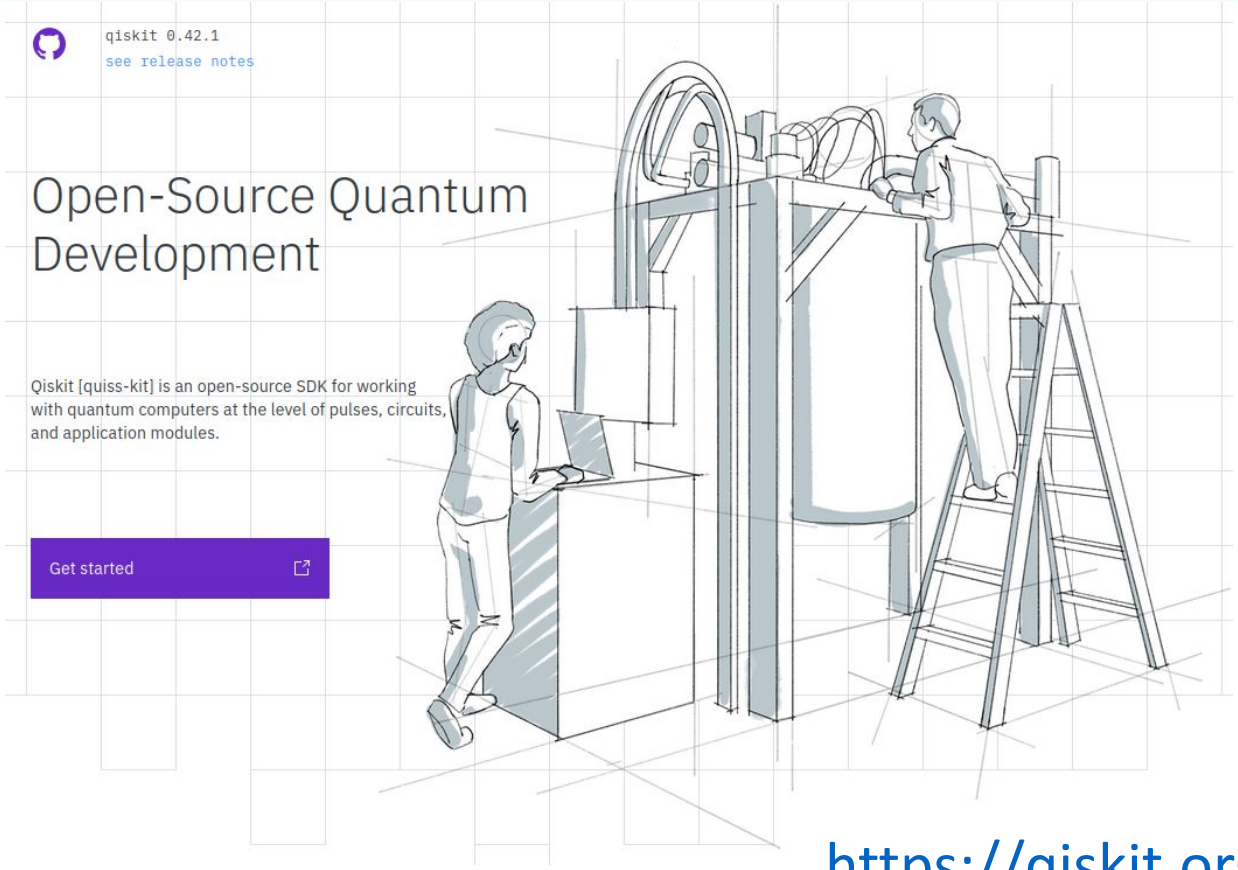
Shor's algorithm



How do we do quantum computing?



Using Qiskit



https://qiskit.org

Write and Run Quantum code

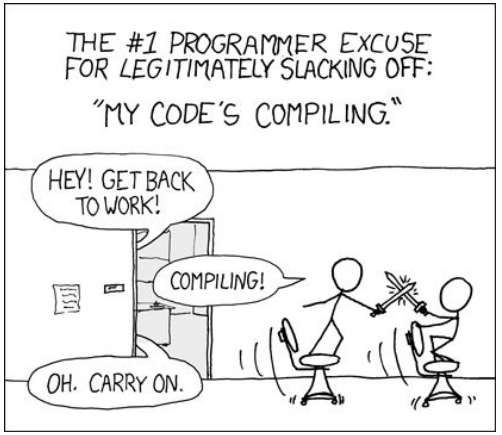
- Tools for composing quantum programs
- Collection of quantum algorithms
- Applications for industry use cases:
  - Machine Learning
  - Nature
  - Finance
  - Optimizations
- Simulate quantum hardware
- Run on real quantum hardware

Four stage workflow

- Build

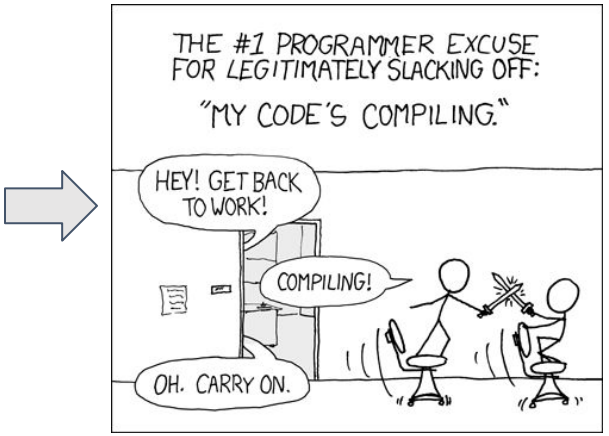
Four stage workflow

- Build
- Compile



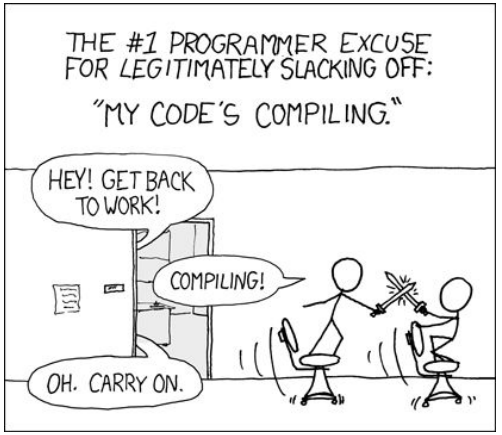
Four stage workflow

- Build
- Compile
- Run



Four stage workflow

- Build
- Compile
- Run
- Analyze



(demo) Qiskit

KubeCon | CloudNativeCon Europe 2023

Build

Now, let's build our quantum circuit.

In [2]:
```
# Create a Quantum Circuit acting on the q register
circuit = QuantumCircuit(2, 2)
```

Here we're initializing a quantum circuit `circuit` with 2 qubits in the zero state and with 2 classical bits set to zero

Next, we'll add a few gates to the circuit

In [3]:
```
# Add a H gate on qubit 0
circuit.h(0)

# Add a CX (CNOT) gate on control qubit 0 and target qubit 1
circuit.cx(0, 1)

# Map the quantum measurement to the classical bits
circuit.measure([0, 1], [0, 1])
```

Out[3]: <qiskit.circuit.instructionset.InstructionSet at 0x7f[illegible]0cca9[illegible]0>

(demo)

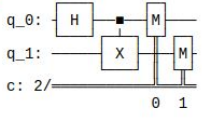
The code above applies the following gates:

- `QuantumCircuit.h(0)` : A Hadamard gate `H` on qubit 0, which puts it into a **superposition state**.
- `QuantumCircuit.cx(0, 1)` : A controlled-Not operation (`CNOT`) on control qubit 0 and target qubit 1, putting the qubits in an **entangled state**.
- `QuantumCircuit.measure([0,1], [0,1])` : If you pass the entire quantum and classical registers to `measure`, the ith qubit's measurement result will be stored in the ith classical bit.

To double-check what we've done, we can view the circuit

In [4]:
```
# Draw the circuit
circuit.draw()
```

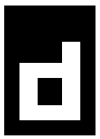
Out[4]:



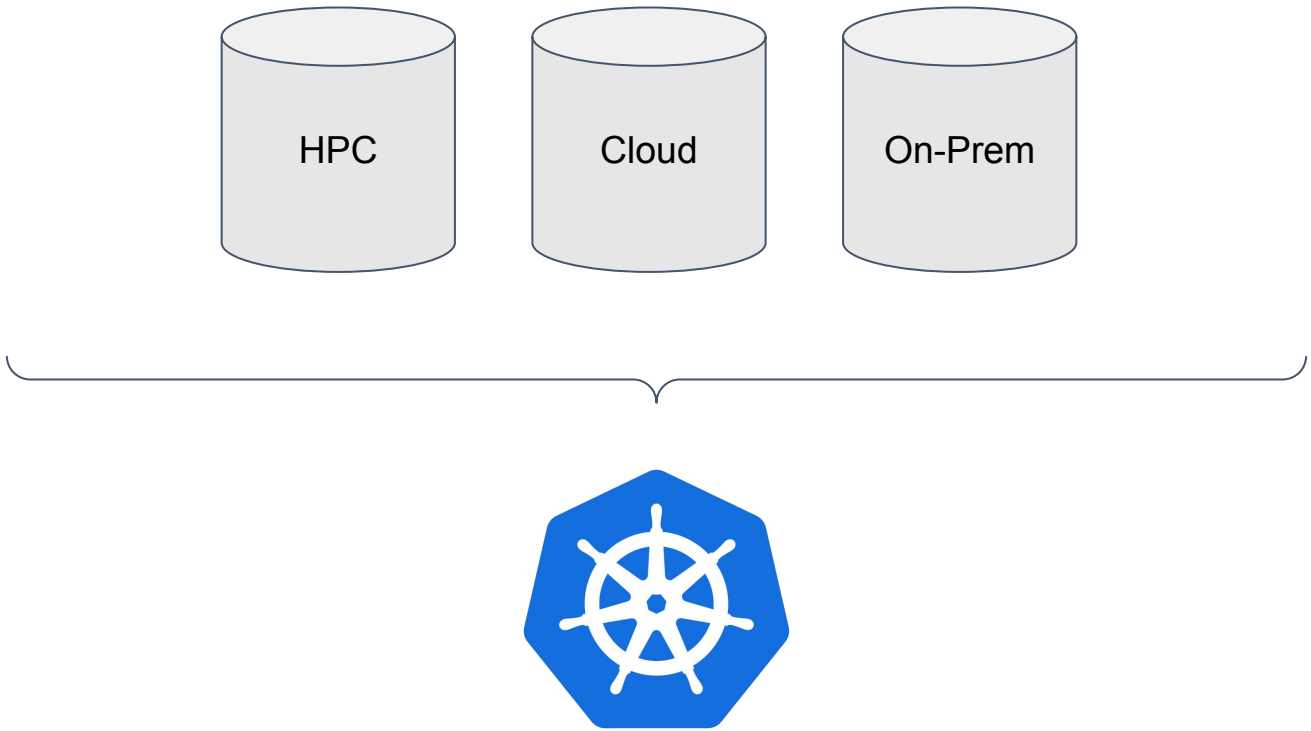
In this circuit, the qubits are ordered with qubit zero at the top and qubit one at the bottom. The circuit is read left to right, meaning that gates which are

How do we do cloud-native quantum computing?

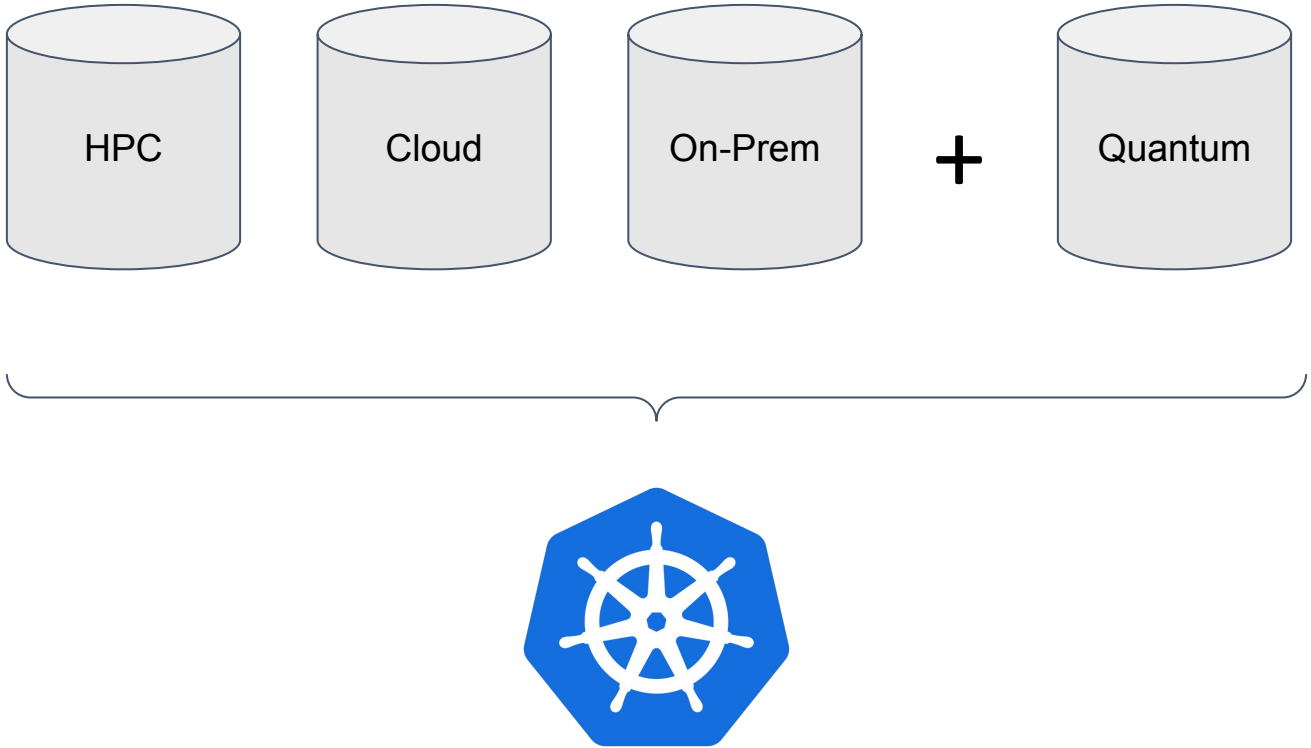


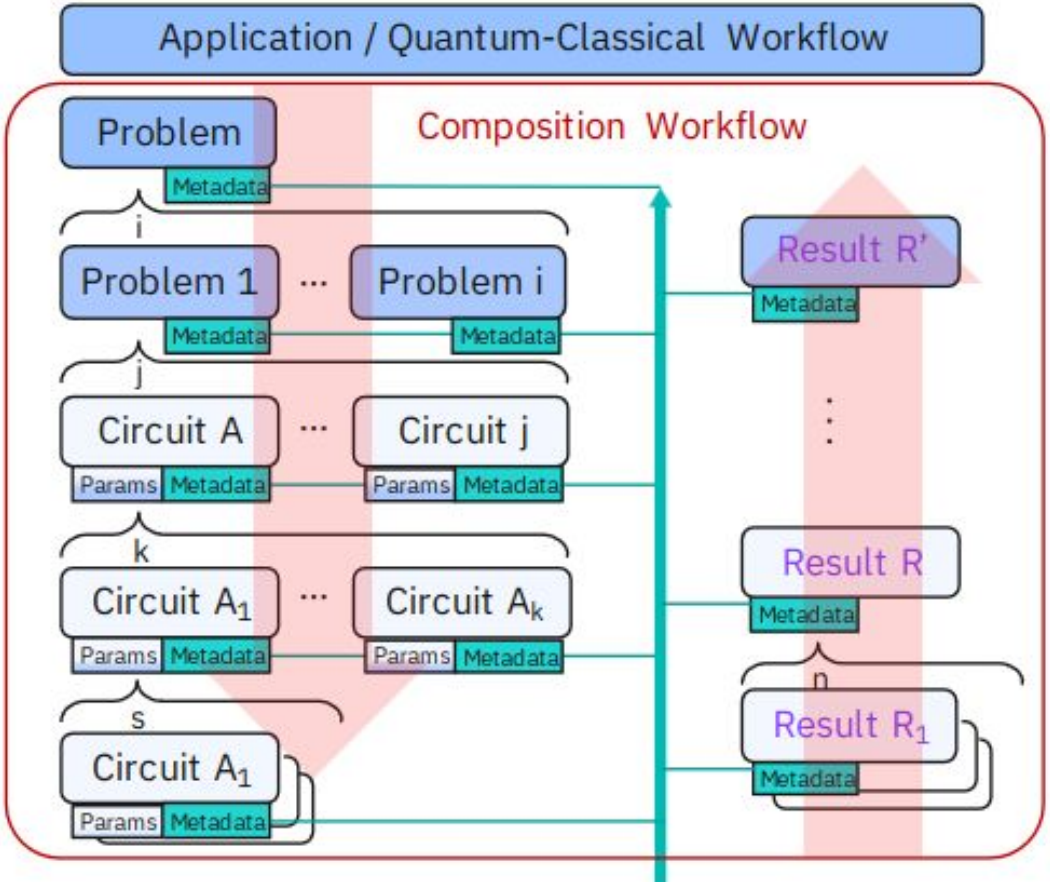
Orchestration by Kubernetes



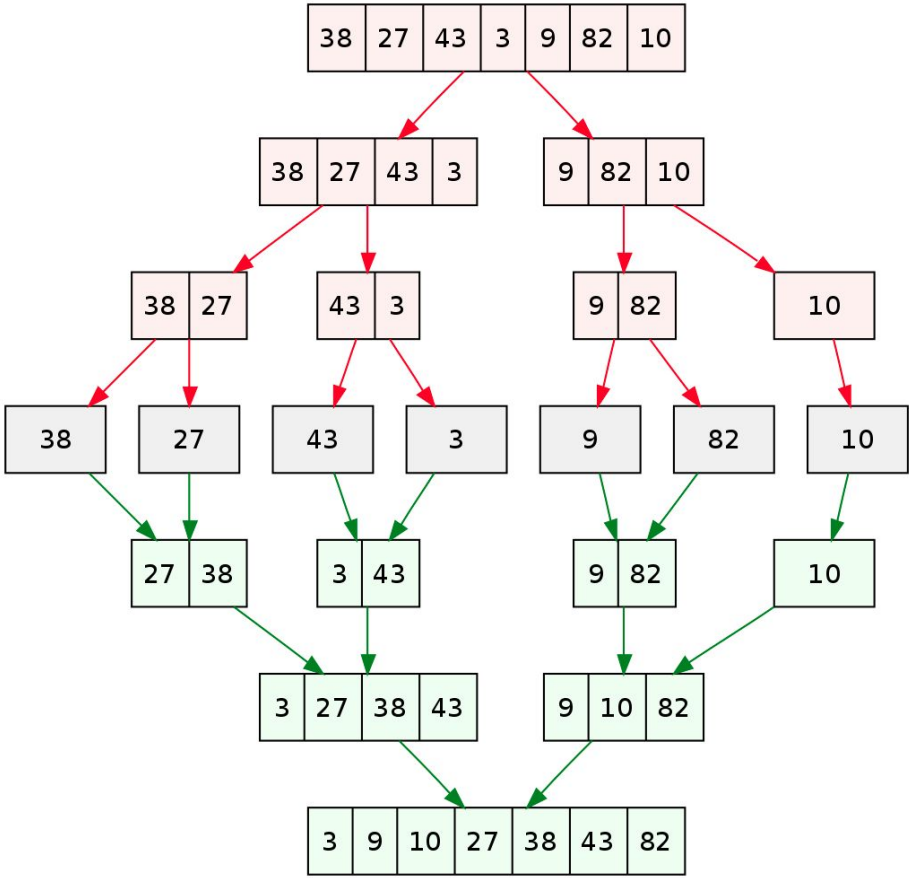
Orchestration by Kubernetes



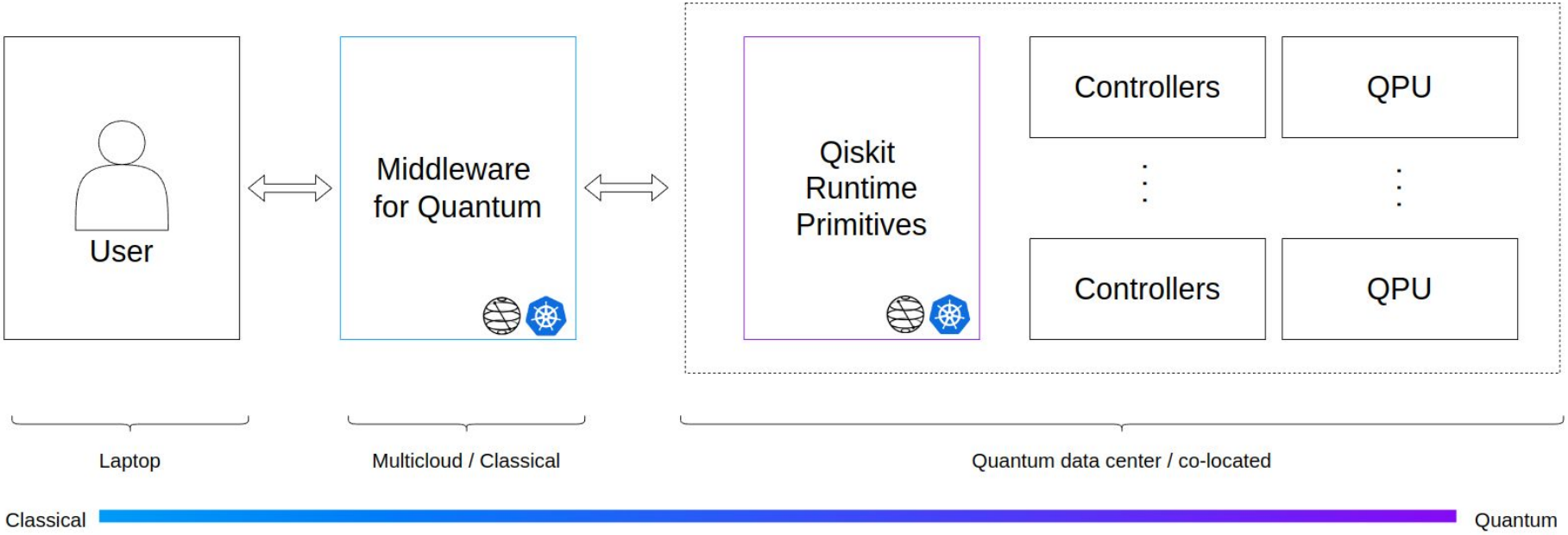
A more complex example…



(divide and conquer)



Quantum Serverless: Classical + Quantum



https://github.com/Qiskit-Extensions/quantum-serverless

Ex: Ground state energy of LiH

Calculate energy

$$E = \sum_i c_i \langle \Psi(\theta) | P_i | \Psi(\theta) \rangle \geq E_{\text{exact}}$$

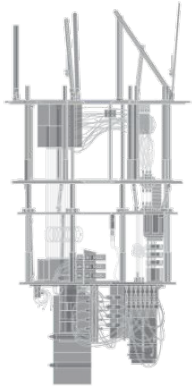
Adjust parameters

$$\{\theta_i\}$$

parameters

Optimize

energy



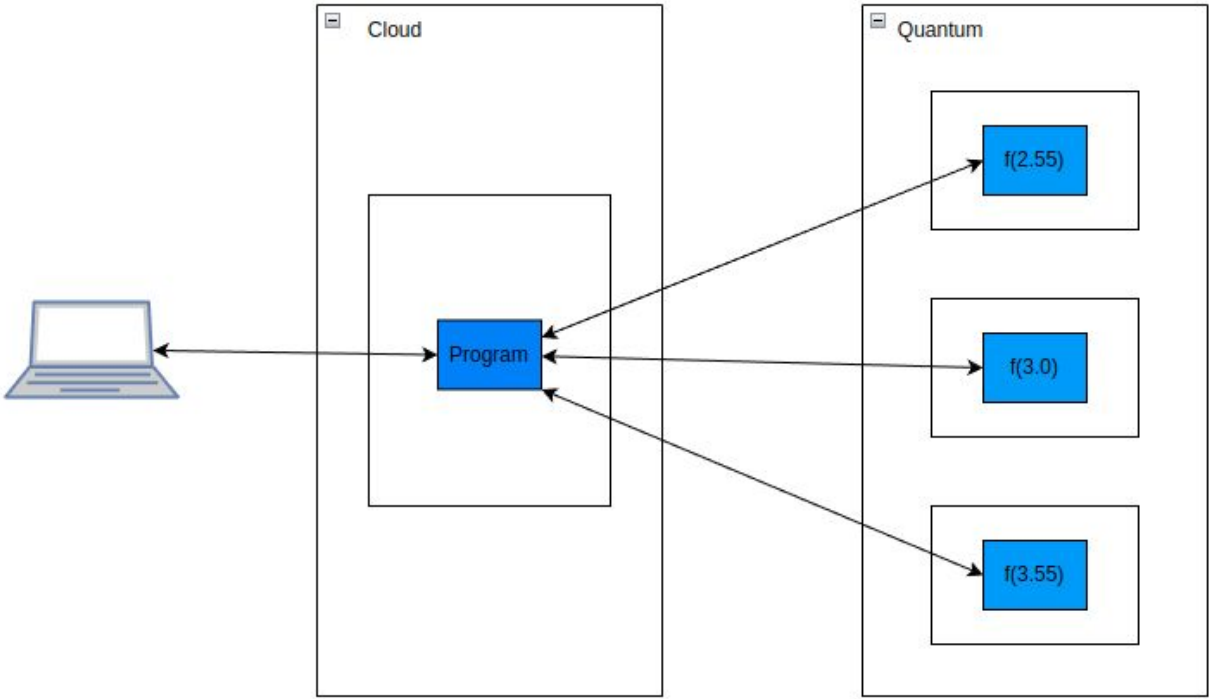
Prepare trial state

$$|\Psi(\theta)\rangle$$

Measure expectation values

$$\langle \Psi(\theta) | P_i | \Psi(\theta) \rangle$$

Parallelize workloads



(demo 2) quantum serverless

KubeCon CloudNativeCon Europe 2023

```
jobs = []

for bond_length in [2.55, 3.0, 3.55]:
    program = Program(
        title=f"Groundstate with bond length {bond_length}",
        entrypoint="gs_level_2.py",
        working_dir="./source_files",
        dependencies=["qiskit-nature", "qiskit-nature[pyscf]"],
        arguments={
            "bond_length": bond_length
        }
    )
    jobs.append(serverless.run_program(program))

jobs
```

```
[<Job | ddb7331c-6db2-4e44-9449-b58ba53bd[illegible]b8>,
 <Job | 52f77d9d-c669-4526-8b33-2fa9290b[illegible]dd>,
 <Job | c31428f3-8c15-4257-a53d-fd[illegible]e2[illegible]>]
```

```
for job in jobs:
    print(job.status())
```

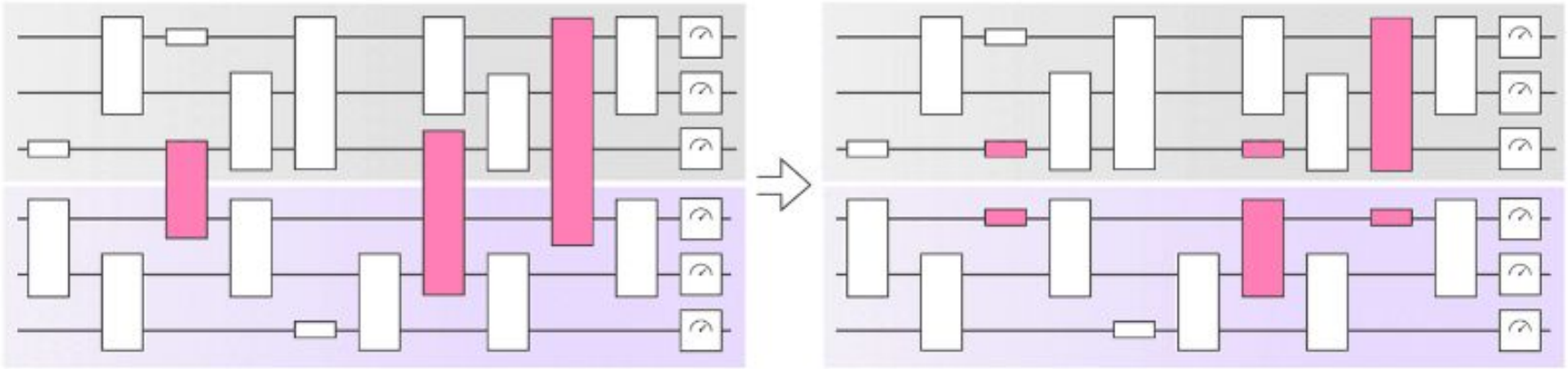
```
RUNNING
RUNNING
RUNNING
```

```
for job in jobs:
    print(job.logs())
```

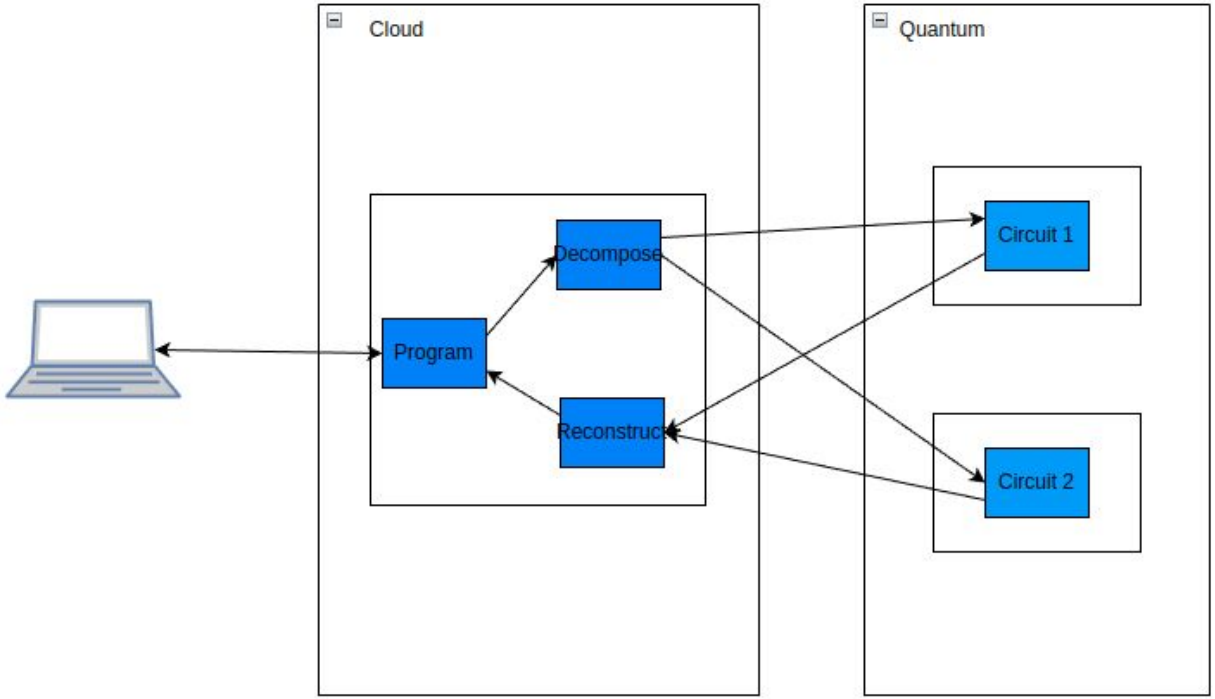
```
Running for bond length 2.55.
=== GROUND STATE ENERGY ===

* Electronic ground state energy (Hartree): -8.211426461751
  - computed part:      -8.211426461751
  - ActiveSpaceTransformer extracted energy part: 0.0
~ Nuclear repulsion energy (Hartree): 0.622561424612
> Total ground state energy (Hartree): -7.588865037139

=== MEASURED OBSERVABLES ===

  0:  # Particles: 3.997 S: 0.436 S^2: 0.626 M: 0.001
```

(demo)

Ex: Circuit Knitting



Decompose / evaluate / reconstruct



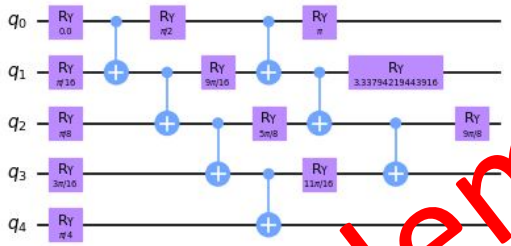
(demo 3) circuit cutting

KubeCon CloudNativeCon Europe 2023

The two subcircuits produced

In [5]:
```
# visualize the first subcircuit
cuts["subcircuits"][0].draw("mpl", fold=-1, scale=0.6)
```

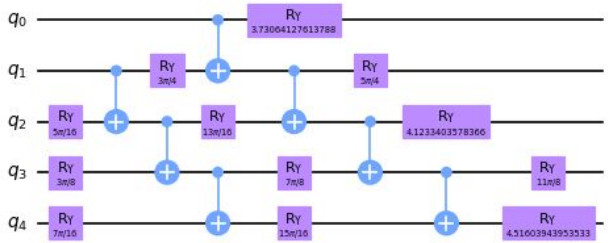
Out[5]:



(demo)

In [6]:
```
# visualize the second subcircuit
cuts["subcircuits"][1].draw("mpl", fold=-1, scale=0.6)
```

Out[6]:



In conclusion…

Cloud-native quantum

- Quantum can solve some “hard” problems
- Quantum + Classical to solve bigger problems
- Kubernetes is a natural orchestrator

Learn more

KubeCon CloudNativeCon Europe 2023

Quantum computing

Qiskit

Quantum Serverless

KubeCon CloudNativeCon Europe 2023

KubeCon CloudNativeCon North America 2021

Please scan the QR Code above
to leave feedback on this session


KubeCon
CloudNativeCon
Europe 2023
TiKV

Backup

IBM Development Roadmap