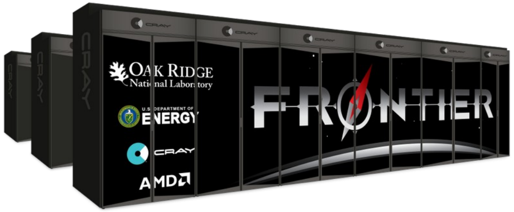
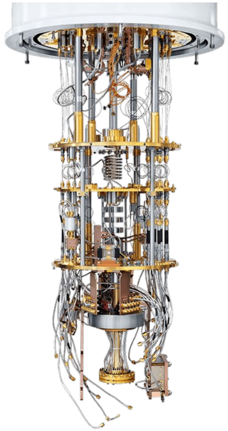
Quantum Hardware in HPC Centers: Integration and Performance Benchmarking and Profiling



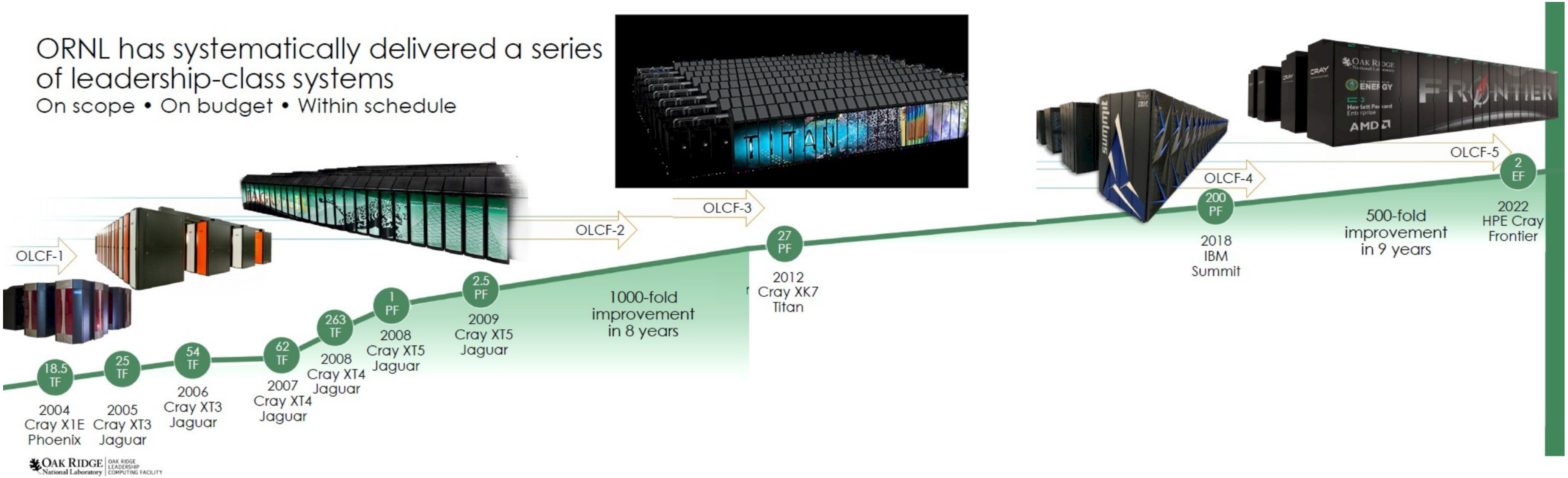
Peter Groszkowski



OLCF Users Conference Call
April 2025

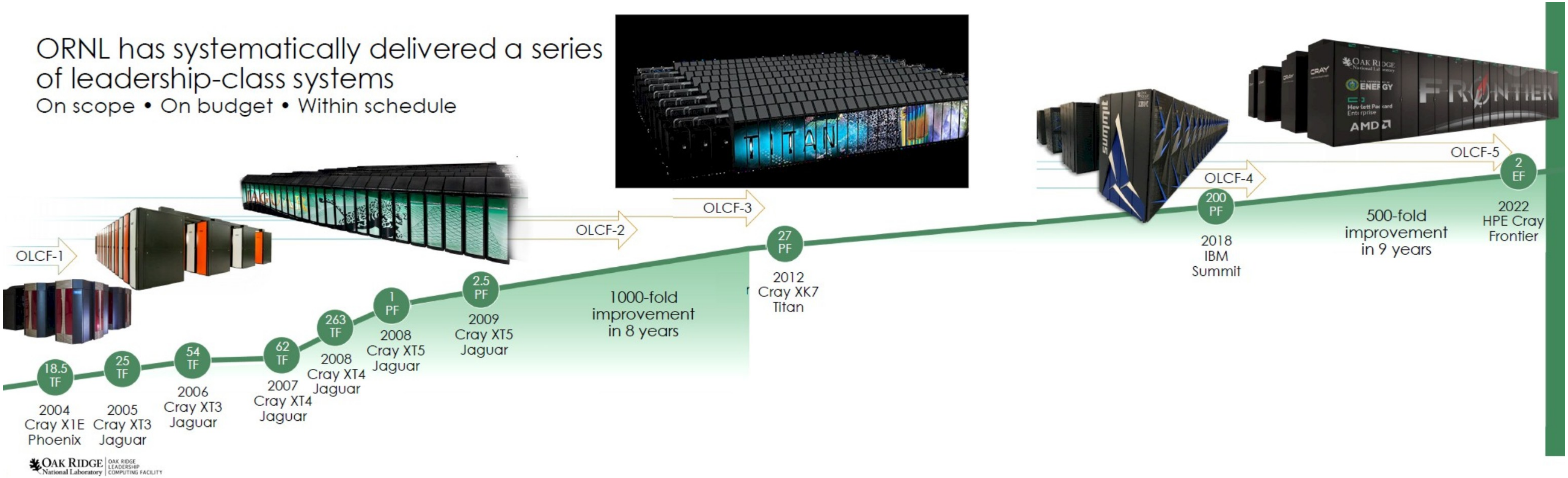
OAK RIDGE
National Laboratory

High Performance Computing at ORNL



- Long history in realizing **leadership scale** HPC systems
- One of the first ones to integrate GPUs into the mix(!) → paradigm shift!
- OLCF6: in planning stages

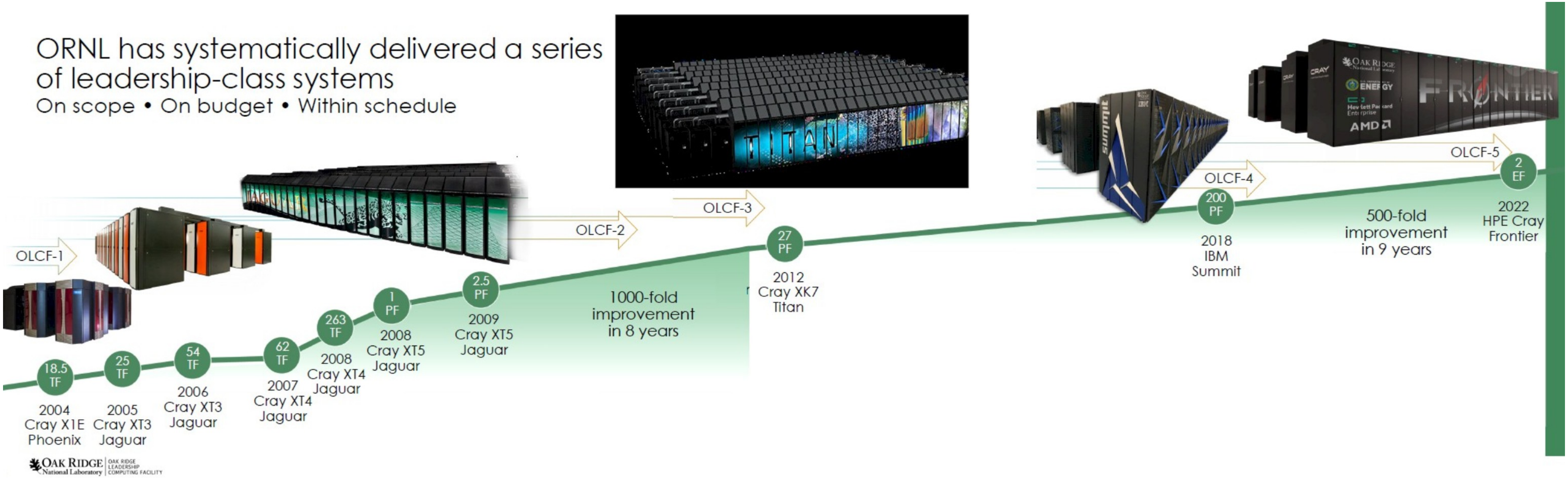
High Performance Computing at ORNL



- Long history in realizing **leadership scale** HPC systems
- One of the first ones to integrate GPUs into the mix(!) → paradigm shift!
- OLCF6: in planning stages

But getting harder to be cost effective (in terms of money, power)

High Performance Computing at ORNL



- Long history in realizing **leadership scale** HPC systems
- One of the first ones to integrate GPUs into the mix(!) → paradigm shift!
- OLCF6: in planning stages

What's next!?… Quantum?

Outline

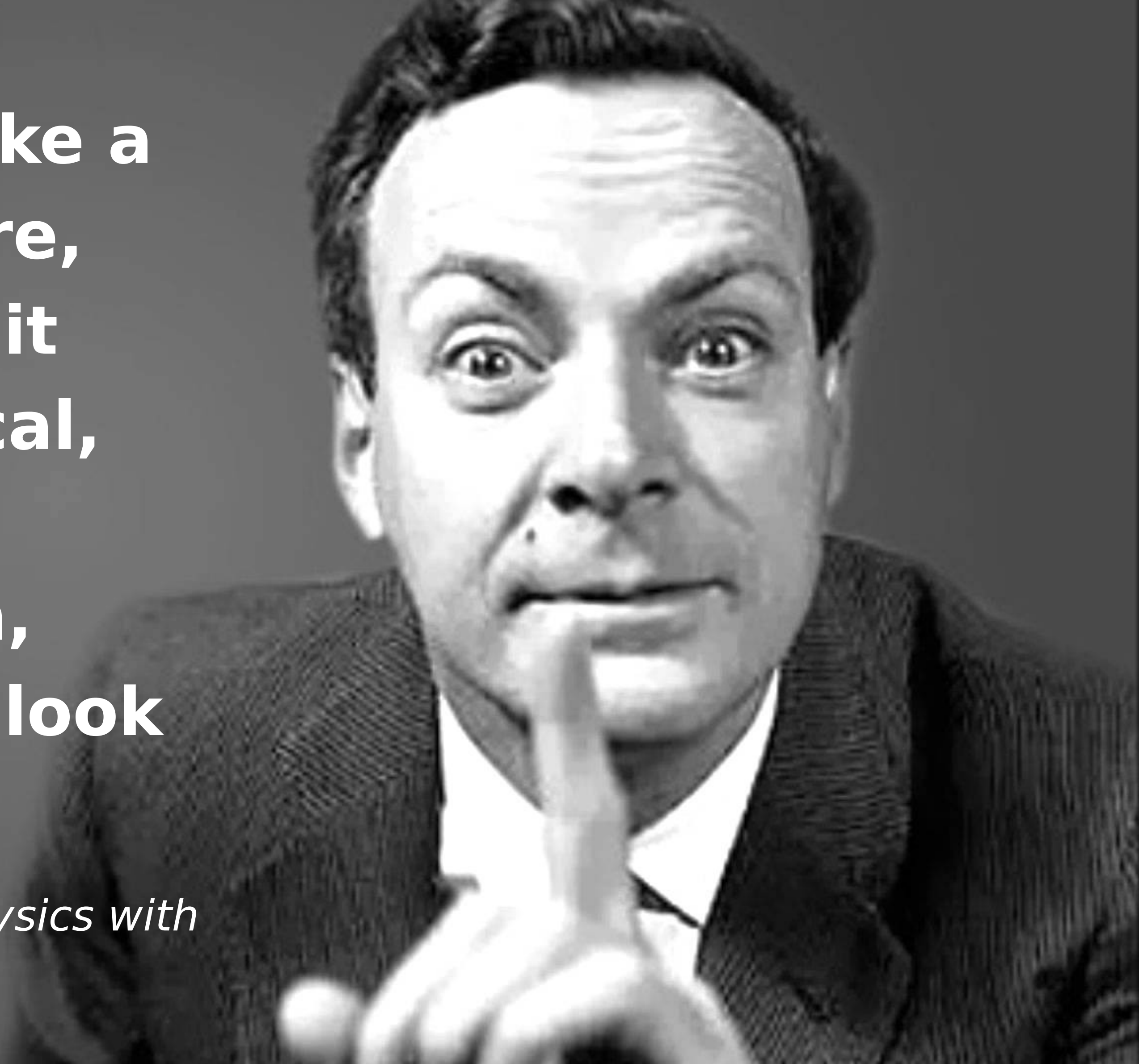
- Quantum Computing 101
- Integration of Quantum Hardware into HPC centers
- Benchmarking **joint** Quantum / HPC systems with **QStone**

Outline

- Quantum Computing 101
- Integration of Quantum Hardware into HPC centers
- Benchmarking **joint** Quantum / HPC systems with **QStone**

“If you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it’s a wonderful problem, because it doesn't look so easy.”

Richard Feynman, *Simulating Physics with Computers* May 1981



Basics of Quantum Computing

- In Quantum Mechanics a **wave function** (“state”) describes knowledge of a system

- Quantum Computing involves appropriately **manipulating this wave function to do computation** while taking advantage of quantum “effects”:

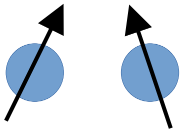
  (1) entanglement: **correlations** between different parts of quantum system

Basics of Quantum Computing

- In Quantum Mechanics a **wave function** (“state”) describes knowledge of a system

- Quantum Computing involves appropriately **manipulating this wave function to do computation** while taking advantage of quantum “effects”:

(1) entanglement: **correlations** between different parts of quantum system



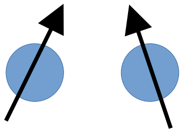
Entangled electrons

Basics of Quantum Computing

- In Quantum Mechanics a **wave function** (“state”) describes knowledge of a system

- Quantum Computing involves appropriately **manipulating this wave function to do computation** while taking advantage of quantum “effects”:

  (1) entanglement: **correlations** between different parts of quantum system



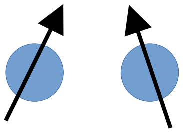
Entangled electrons

One moved “far away”

Basics of Quantum Computing

- In Quantum Mechanics a **wave function** (“state”) describes knowledge of a system
- Quantum Computing involves appropriately **manipulating this wave function to do computation** while taking advantage of quantum “effects”:

(1) entanglement: **correlations** between different parts of quantum system



Entangled electrons

One moved “far away”

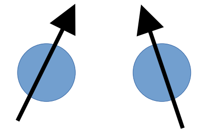
Measure outcome of first determines state of second(!)

Basics of Quantum Computing

- In Quantum Mechanics a **wave function** (“state”) describes knowledge of a system

- Quantum Computing involves appropriately **manipulating this wave function to do computation** while taking advantage of quantum “effects”:

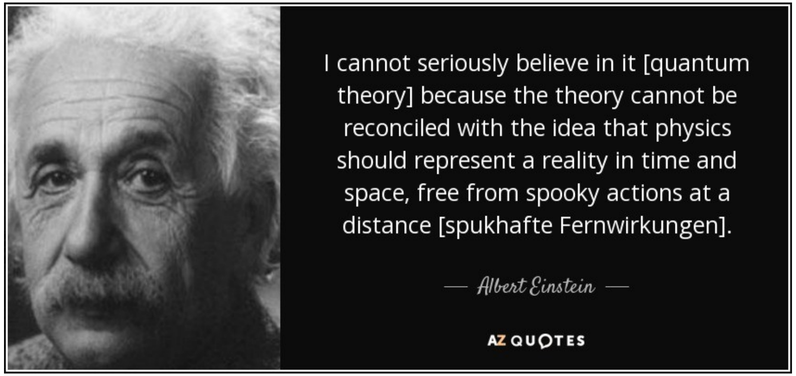
(1) entanglement: **correlations** between different parts of quantum system



Entangled electrons

One moved “far away”

Measure outcome of first determines state of second(!)



Basics of Quantum Computing

- In Quantum Mechanics a **wave function** (“state”) describes knowledge of a system

- Quantum Computing involves appropriately **manipulating this wave function to do computation** while taking advantage of quantum “effects”:

  (1) entanglement: **correlations** between different parts of quantum system

  (2) superposition: state that captures **multiple** possible system configurations

Basics of Quantum Computing

- In Quantum Mechanics a **wave function** (“state”) describes knowledge of a system
- Quantum Computing involves appropriately **manipulating this wave function to do computation** while taking advantage of quantum “effects”:

(1) entanglement: **correlations** between different parts of quantum system

(2) superposition: state that captures **multiple** possible system configurations

Classical Bits: 0 OR 1

Basics of Quantum Computing

- In Quantum Mechanics a **wave function** (“state”) describes knowledge of a system

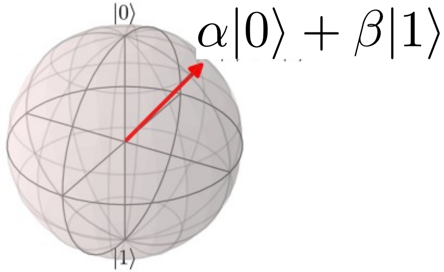
- Quantum Computing involves appropriately **manipulating this wave function to do computation** while taking advantage of quantum “effects”:

  (1) entanglement: **correlations** between different parts of quantum system

  (2) superposition: state that captures **multiple** possible system configurations

Classical Bits: 0 OR 1

Quantum Bits (qubits):



Basics of Quantum Computing

- In Quantum Mechanics a **wave function** (“state”) describes knowledge of a system

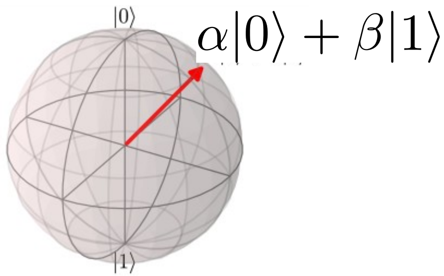
- Quantum Computing involves appropriately **manipulating this wave function to do computation** while taking advantage of quantum “effects”:

  (1) entanglement: **correlations** between different parts of quantum system

  (2) superposition: state that captures **multiple** possible system configurations

Classical Bits: 0 OR 1

Quantum Bits (qubits):



- With N qubits, can represent 2^N configurations! (e.g., for N=50 → 1,125,899,906,842,624)
  → very hard to simulate quantum systems on classical computers!

OAK RIDGE National Laboratory

Quantum Algorithms

- Quantum algorithms are defined as sequences of **gates** (think: “simple instructions”)

Quantum Algorithms

- Quantum algorithms are defined as sequences of **gates** (think: “simple instructions”)
- **Gates** are realized by manipulating quantum systems (e.g., with electric and magnetic fields)

Schrödinger equation

$$i\hbar \frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H}_{\mathrm{sys}}(t) |\Psi(t)\rangle$$

quantum state

“Hamiltonian”
(tune to do gates)

Quantum Algorithms

- Quantum algorithms are defined as sequences of **gates** (think: “simple instructions”)
- **Gates** are realized by manipulating quantum systems (e.g., with electric and magnetic fields)

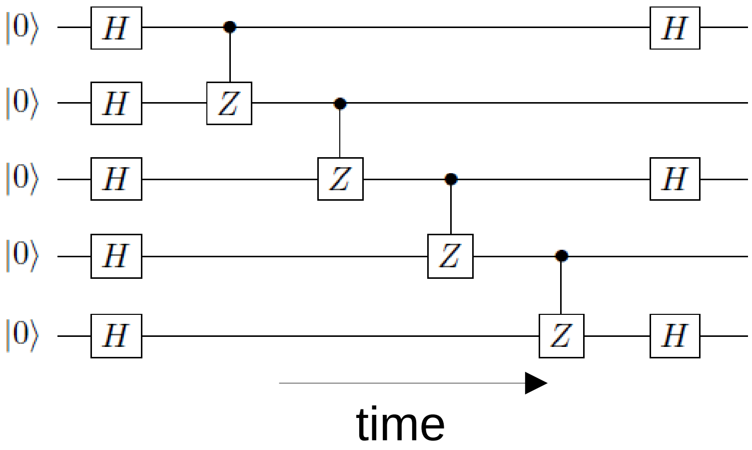
Schrödinger equation

$$i\hbar \frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H}_{\text{sys}}(t)\, |\Psi(t)\rangle$$

quantum state

“Hamiltonian”
(tune to do gates)

Quantum algorithm (circuit)



Quantum Algorithms

- Quantum algorithms are defined as sequences of **gates** (think: “simple instructions”)
- **Gates** are realized by manipulating quantum systems (e.g., with electric and magnetic fields)

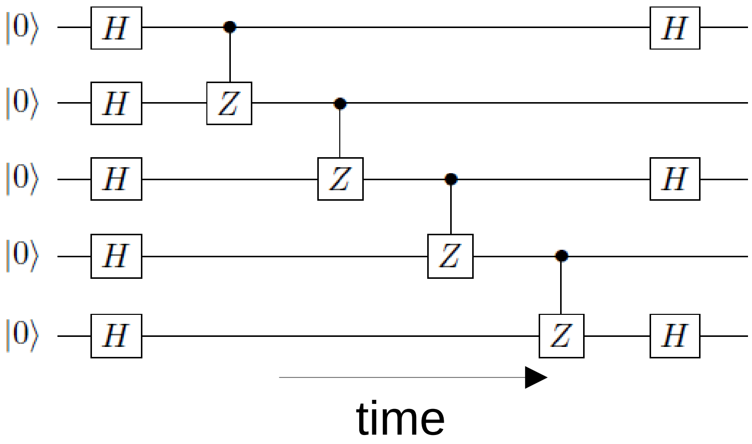
Schrödinger equation

$$i\hbar \frac{\partial}{\partial t} |\Psi(t)\rangle = \hat{H}_{\text{sys}}(t)\, |\Psi(t)\rangle$$

quantum state

“Hamiltonian”
(tune to do gates)

Quantum algorithm (circuit)



- Many algorithms have been proposed related to:

  factoring, searching, quantum simulation, machine learning, solving linear equations

How to Build a Quantum Computer?

- “Just” need a “well behaved” quantum system!

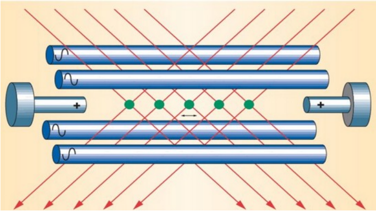
That we can:

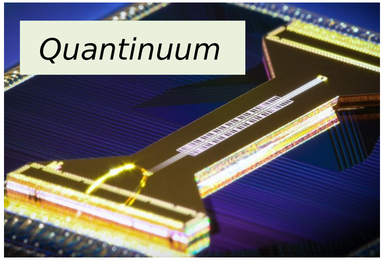
- Initialize
- Measure
- Control
- … that will stay “quantum”

How to Build a Quantum Computer?

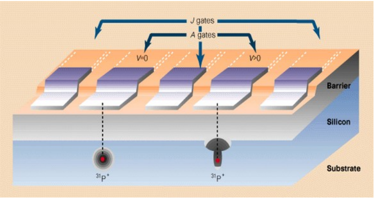
- "Just" need a "well behaved" quantum system!

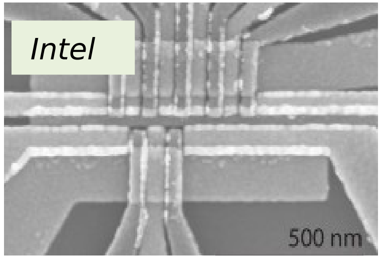
Trapped Ions

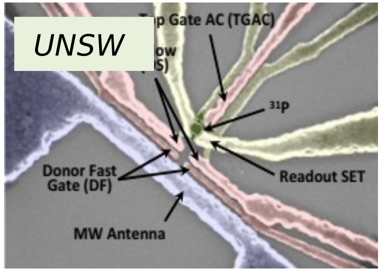




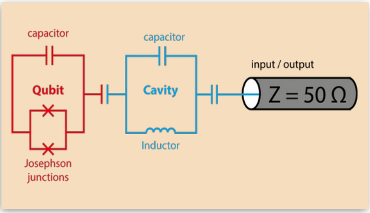
Spin qubits

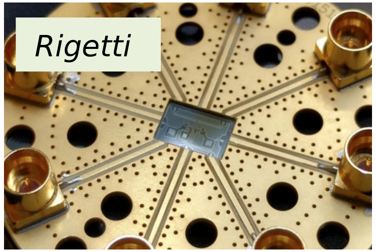


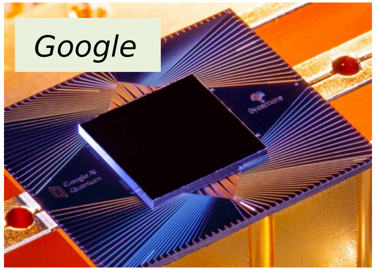




Superconducting circuits



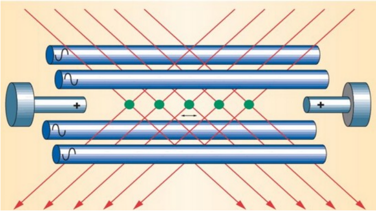


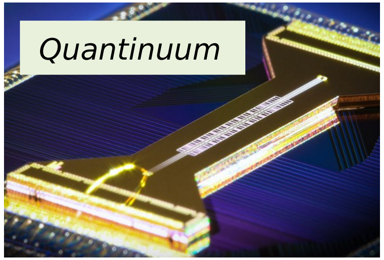


How to Build a Quantum Computer?

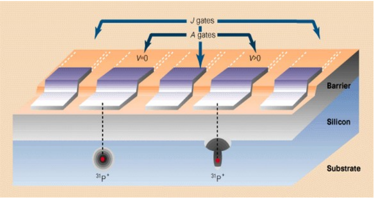
- “Just” need a “well behaved” quantum system!

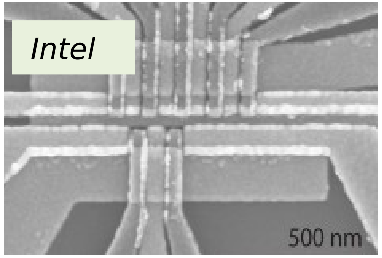
Trapped Ions

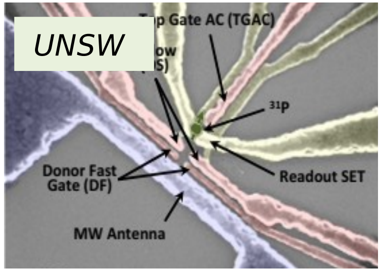




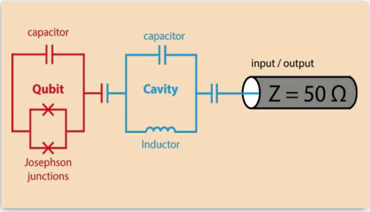
Spin qubits

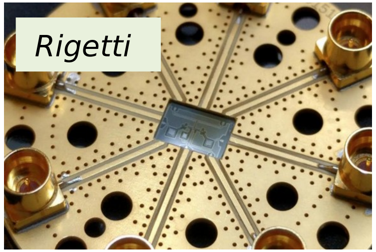


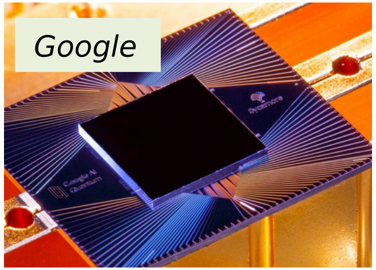




Superconducting circuits







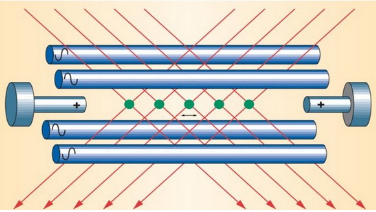
Other approaches include:

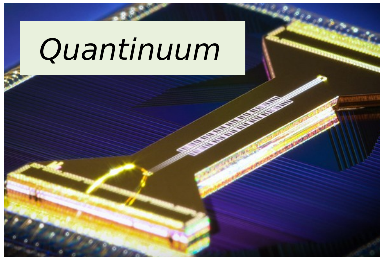
- Photonic qubits (PsiQuantum, Xanadu)
- Topological qubits (Microsoft, Delft)
- Other ideas also exist

How to Build a Quantum Computer?

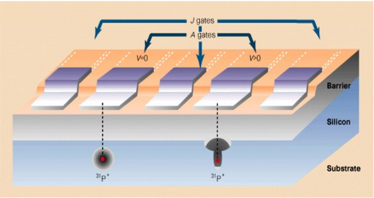
- “Just” need a “well behaved” quantum system!

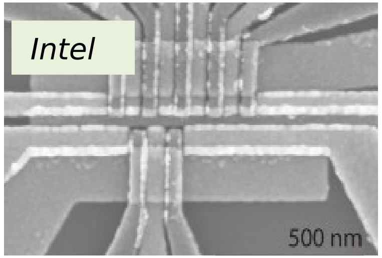
Trapped Ions

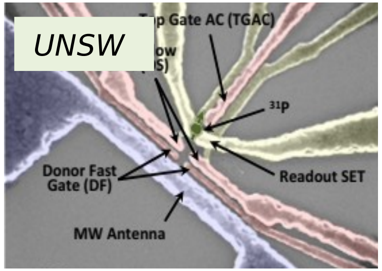




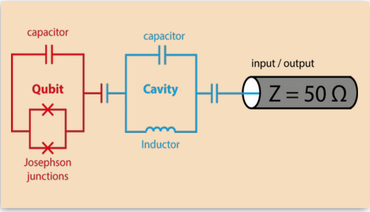
Spin qubits

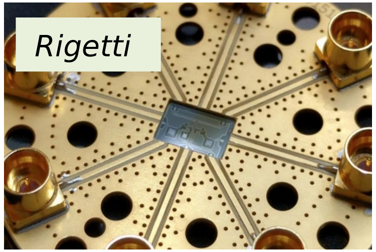


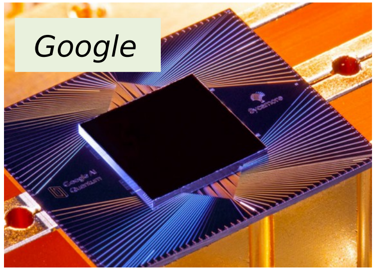




Superconducting circuits







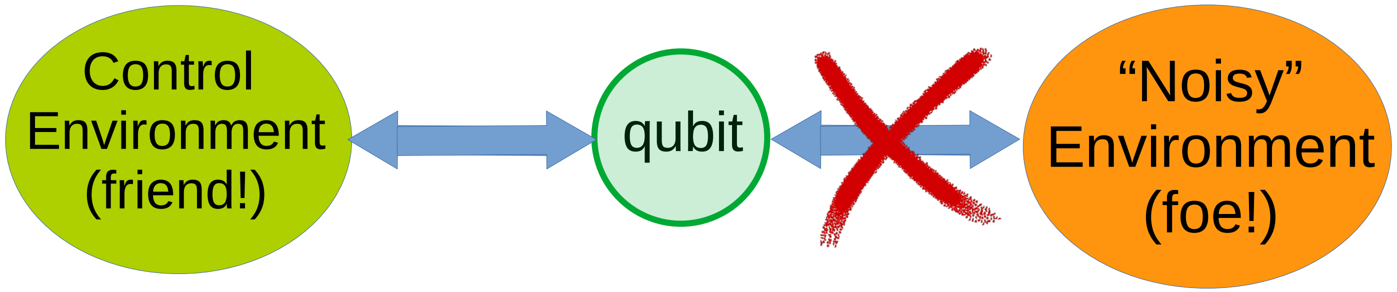
Other approaches include:

- Photonic qubits (PsiQuantum, Xanadu)
- Topological qubits (Microsoft, Delft)
- Other ideas also exist

- **Too early to know what technology will succeed**
- **Different trade-offs between systems**

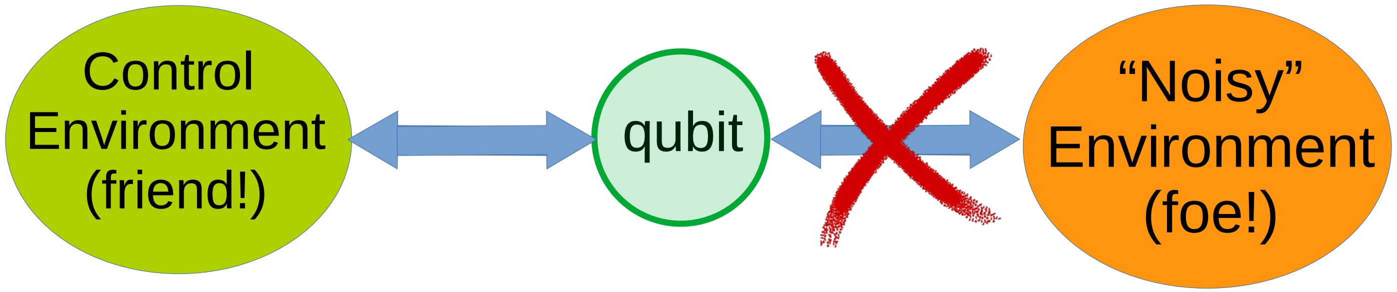
Why are Quantum Computers so Hard to Build?

- We need systems that are **isolated** from the environment (to limit effects of **"bad"** noise),
- … but ones that we can **control...** **all at the same time!**



Why are Quantum Computers so Hard to Build?

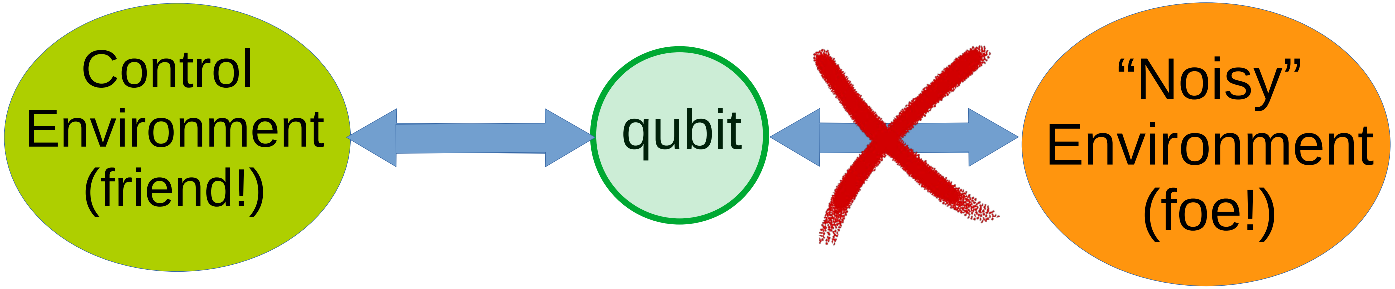
- We need systems that are **isolated** from the environment (to limit effects of **“bad”** noise),
- … but ones that we can **control...** **all at the same time!**



- Very hard to do – can’t (easily) have one without the other!

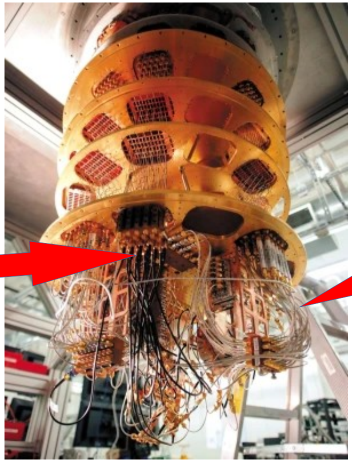
Why are Quantum Computers so Hard to Build?

- We need systems that are **isolated** from the environment (to limit effects of **“bad”** noise),
- … but ones that we can **control...** **all at the same time!**



- Very hard to do – can’t (easily) have one without the other!

Small quantum computer



Cables to classical electronics

Allow for control,
but also bring noise

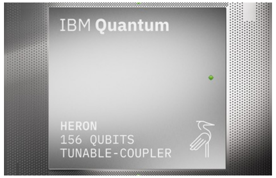
[Mohseni et al., Nature (2017)]

Noisy Intermediate-Scale Quantum (NISQ) Era

- Current state of the art:

  algorithms that are **10s of gates** “deep” on **small & noisy** **~100 qubit machines**

- Too limited for “large-scale quantum” algorithms (e.g. Shor’s factoring)



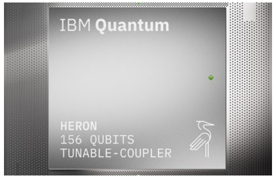
IBM: “Heron”, 156 qubits

Noisy Intermediate-Scale Quantum (NISQ) Era

- Current state of the art:

  algorithms that are **10s of gates** "deep" on **small & noisy** **~100 qubit machines**

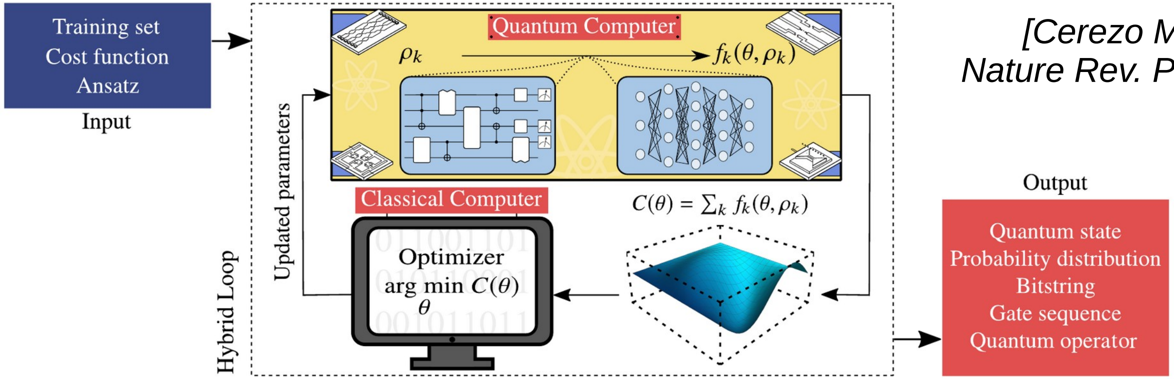
- Too limited for "large-scale quantum" algorithms (e.g. Shor's factoring)



IBM: "Heron", 156 qubits

- Very active research area for NISQ devices: classical/quantum **hybrid variational algorithms**:

  Use classical optimization **together** with small quantum circuits



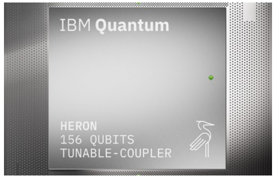
[Cerezo M. et al., Nature Rev. Phys. (2021)]

Noisy Intermediate-Scale Quantum (NISQ) Era

- Current state of the art:

  algorithms that are **10s of gates** “deep” on **small & noisy** **~100 qubit machines**

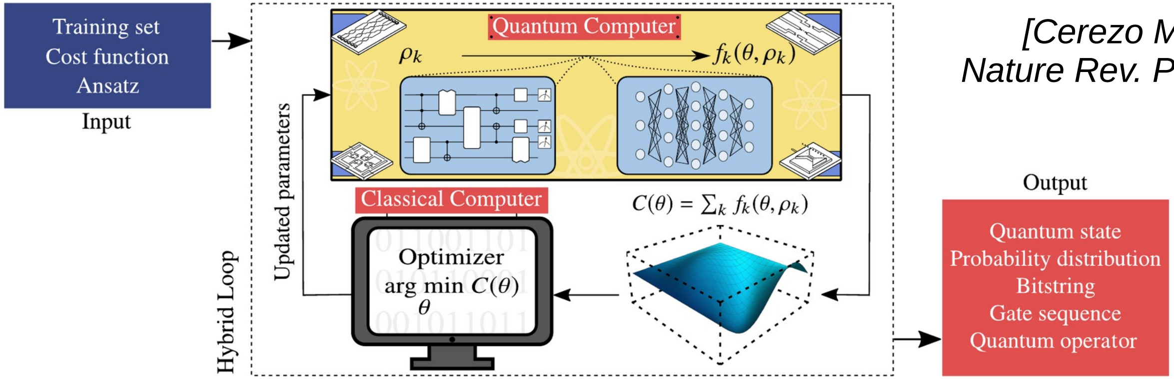
- Too limited for “large-scale quantum” algorithms (e.g. Shor’s factoring)



IBM: “Heron”, 156 qubits

- Very active research area for NISQ devices: classical/quantum **hybrid variational algorithms**:

  Use classical optimization **together** with small quantum circuits



[Cerezo M. et al., Nature Rev. Phys. (2021)]

- Used to explore **small-scale** chemistry simulations and in quantum machine learning and classification

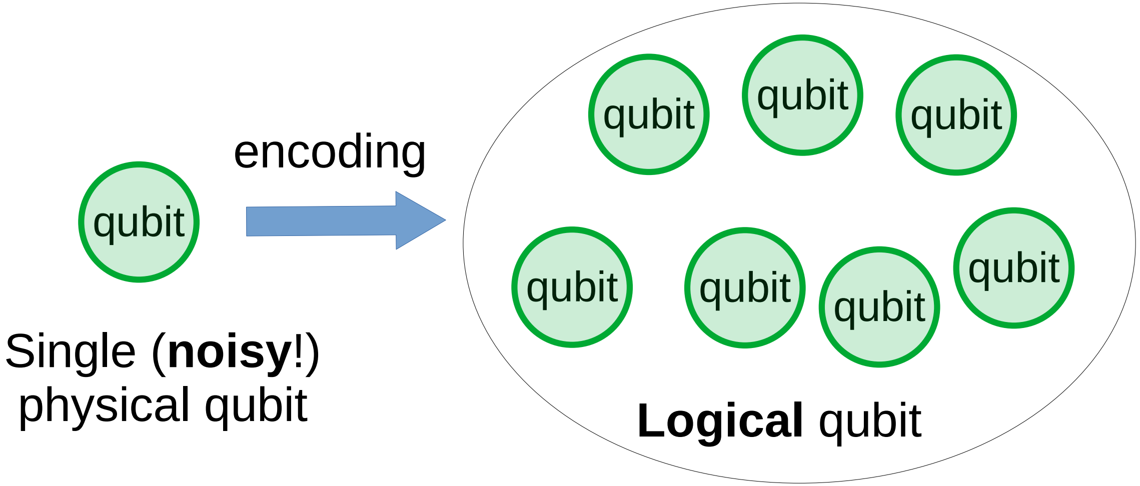
Oak Ridge National Laboratory

Future of Quantum Computing: Quantum Error Correction

- Most likely some unwanted noise effects will always be present!

Future of Quantum Computing: Quantum Error Correction

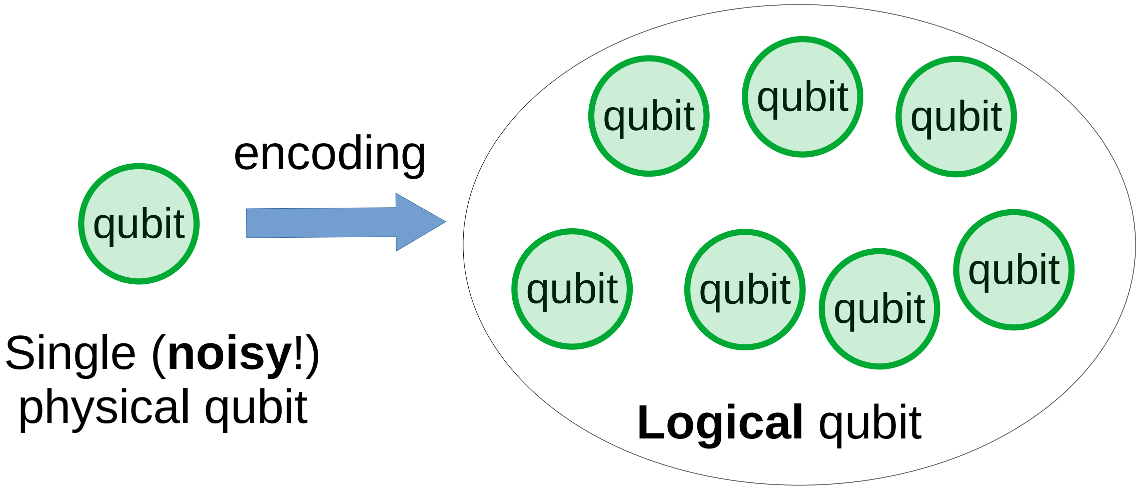
- Most likely some unwanted noise effects will always be present!
- Future machines will need to implement **quantum error correction**



- Now errors can be **corrected** before information is lost

Future of Quantum Computing: Quantum Error Correction

- Most likely some unwanted noise effects will always be present!
- Future machines will need to implement **quantum error correction**



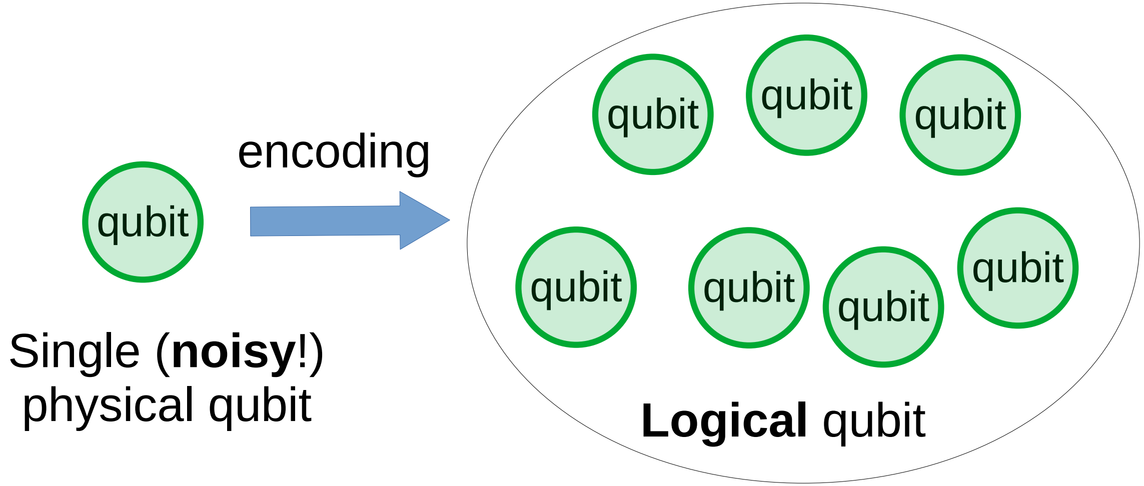
- Now errors can be **corrected** before information is lost

- Initially not obvious that it even be done in quantum systems ("No cloning Theorem")!

  → Peter Shor showed in ~1994, how to do it!

Future of Quantum Computing: Quantum Error Correction

- Most likely some unwanted noise effects will always be present!
- Future machines will need to implement **quantum error correction**



- Now errors can be **corrected** before information is lost

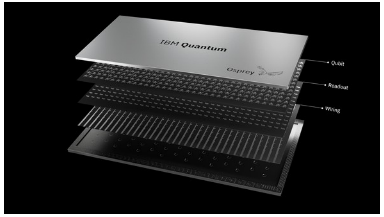
- Initially not obvious that it even be done in quantum systems (“No cloning Theorem”)!
  → Peter Shor showed in ~1994, how to do it!

- Modern estimates show that we may need **millions** of **physical qubits** to run useful algorithms (e.g. Shor’s factoring) → long road ahead

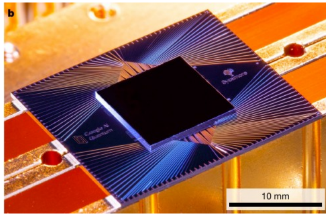
(Selective and Brief) History of Quantum Computing

- Paul Benioff (1979): Computation with Hamiltonians
- Feynman (1981): Simulations
- **Shor**, Grover, Simon, etc. (>1994): Algorithms (e.g. factoring), Quantum Error Correction
- (>1997): Implementations
  - (<2015): noisy 1 to few qubit devices
  - (~2023): noisy 100-400 qubit devices
- "NISQ" Era (>2018): small/noisy algorithms
- Google (2019): "Quantum Supremacy" **experiment**
- Google/Quantinuum (2024): Early "logical qubit" demonstrations

Peter Shor



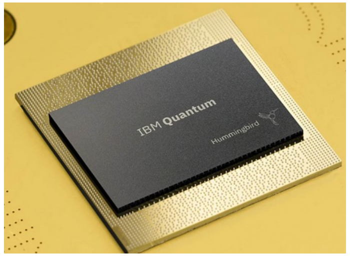
IBM: "Osprey", 433 qubits



Google: 53 qubit "Sycamore" [Nature 574 (2019)]

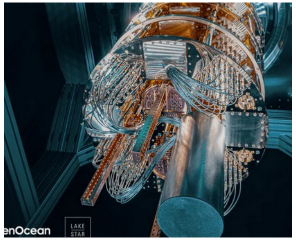
ORNL Quantum Computing User Program (**QCUP**)

Premium access to current stack of **quantum** devices also available through **OLCF**

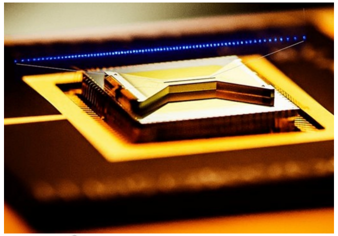


IBM Quantum

QUANTINUUM



IQM



IONQ

- Access to **premium** machines from various vendors
- Support for a broad spectrum of research topics
- **Could involve tool-development**
- Each project gets a “liaison” (ORNL point of contact with quantum science expertise)

Info: https://www.olcf.ornl.gov/

Contact: groszkowskip@ornl.gov

Outline

- Quantum Computing 101
- Integration of Quantum Hardware into HPC centers
- Benchmarking **joint** Quantum / HPC systems with **QStone**

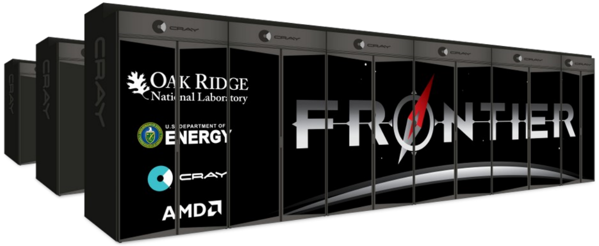
- Integration of Quantum Hardware into HPC centers

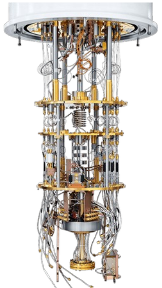
with: Amir Shehata, Thomas Naughton, Daniel Claudino,Thomas Beck

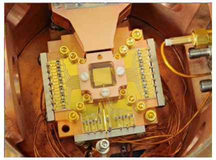
Ongoing Integration Efforts Around the World

- Quantum won’t replace classical (!)
- Likely will work in tandem; **quantum as “accelerators”** (similar to GPUs)
- How to “best” integrate quantum with classical compute?

Classical hardware





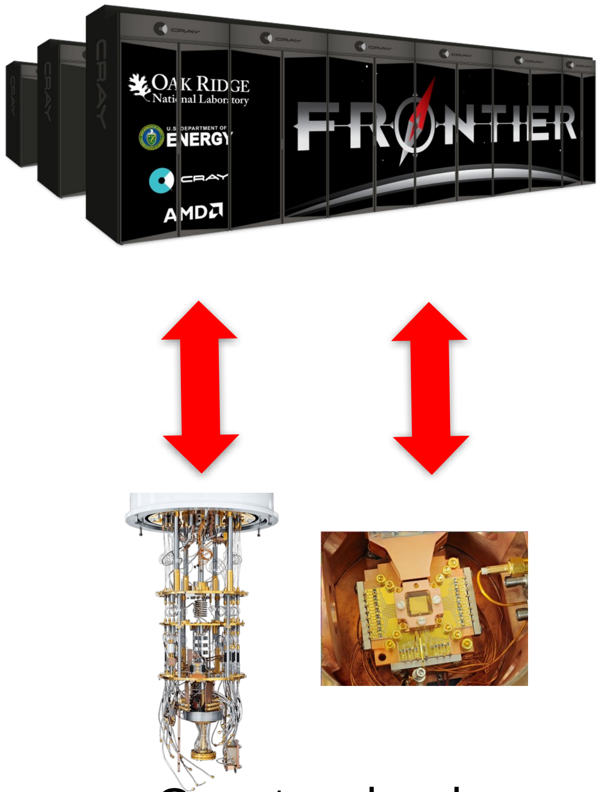


Quantum hardware

Ongoing Integration Efforts Around the World

- Quantum won't replace classical (!)
- Likely will work in tandem; **quantum as "accelerators"** (similar to GPUs)
- How to "best" integrate quantum with classical compute?
- Major recent effort at HPC centers around the world, e.g., LUMI, Pawsey, Leibniz, RIKEN, … ORNL working actively on this!

Classical hardware

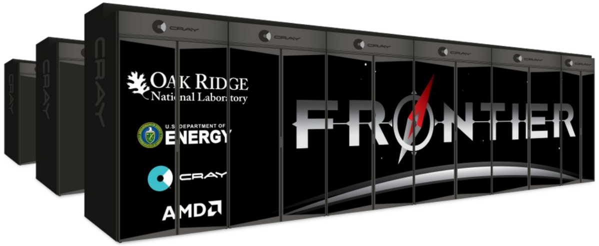


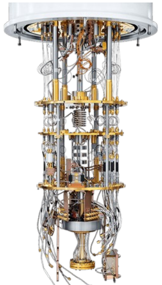
Quantum hardware

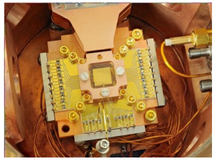
Ongoing Integration Efforts Around the World

- Quantum won't replace classical (!)
- Likely will work in tandem; **quantum as "accelerators"** (similar to GPUs)
- How to "best" integrate quantum with classical compute?
- Major recent effort at HPC centers around the world, e.g., LUMI, Pawsey, Leibniz, RIKEN, … ORNL working actively on this!
- New challenges, e.g.:
  - QPUs may not be local!
  - Variety in hardware architectures
  - Very (!) limited hardware availability (e.g., compare with GPUs)

Classical hardware







Quantum hardware

Promising Hybrid HPC/Quantum Applications?

Lots of classical compute required in quantum… but not clear what's most ideal to take **full advantage** of large **distributed** HPC systems

- Variational Algorithms (e.g. VQE, QML/classification)
- QEC: syndrome decoding (?)
- Transpiling and circuit preprocessing

Promising Hybrid HPC/Quantum Applications?

Lots of classical compute required in quantum… but not clear what's most ideal to take **full advantage** of large **distributed** HPC systems

- Variational Algorithms (e.g. VQE, QML/classification)
- QEC: syndrome decoding (?)
- Transpiling and circuit preprocessing

Heron (IBM)

Fugaku (RIKEN)

Chemistry Beyond Exact Solutions on a Quantum-Centric Supercomputer

Javier Robledo-Moreno,[1,*] Mario Motta,[1,†] Holger Haas,[1] Ali Javadi-Abhari,[1] Petar Jurcevic,[1] William Kirby,[2] Simon Martiel,[3] Kunal Sharma,[1] Sandeep Sharma,[4] Tomonori Shirakawa,[5,6,7] Iskandar Sitdikov,[1] Rong-Yang Sun,[5,6,7] Kevin J. Sung,[1] Maika Takita,[1] Minh C. Tran,[2] Seiji Yunoki,[5,6,7,8] and Antonio Mezzacapo[1,‡]

[1]*IBM Quantum, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598*
[2]*IBM Quantum, IBM Research Cambridge, Cambridge, MA 02142, USA*
[3]*IBM Quantum, IBM France Lab, Orsay, France*
[4]*Department of Chemistry, University of Colorado, Boulder, CO 80302, USA*
[5]*Computational Materials Science Research Team, RIKEN Center for Computational Science (R-CCS), Kobe, Hyogo, 650-0047, Japan*
[6]*Quantum Computational Science Research Team, RIKEN Center for Quantum Computing (RQC), Wako, Saitama, 351-0198, Japan*
[7]*RIKEN Interdisciplinary Theoretical and Mathematical Sciences Program (iTHEMS), Wako, Saitama 351-0198, Japan*
[8]*RIKEN Center for Emergent Matter Science (CEMS), Wako, Saitama 351-0198, Japan*

- Use 6400 Fugaku supercomputer nodes for **selective** diagonalization
- Use 55-77 Heron qubits
- ~3500 2-qubit gates
- No real-time interactions

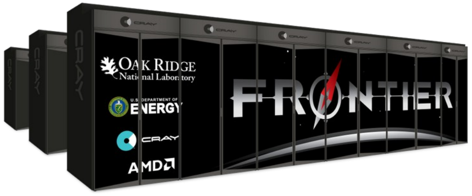
HPC/Quantum Integration: How Will It Work?

Translation layers

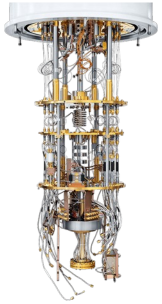
Application
(qiskit, tket, etc)

Middleware
(resource allocation and scheduling circuit processing and translation, classical pre/post processing, etc)

Classical compute
OR simulation of QC



Quantum resource



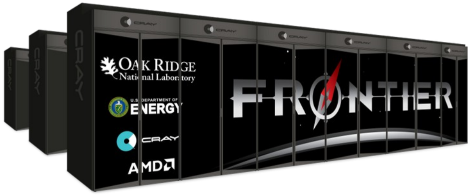
- Critical requirements:
  - Agnostic to the applications and “backends”
  - Quantum simulators first-class citizens (e.g., state vector, tensor network, etc.)
  - Free and open

[Beck at al., Future Generation Computer Systems 161, 11-25 (2024)]

HPC/Quantum Integration: How Will It Work?

Translation layers

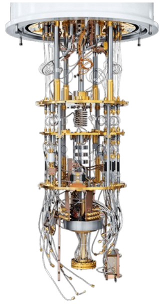
Classical compute
OR simulation of QC



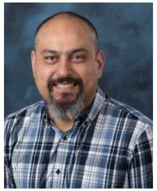
Application
(qiskit, tket, etc)

Middleware
(resource allocation and scheduling circuit processing and translation, classical pre/post processing, etc)

Quantum resource



- Early prototype in place!
- Exploring scheduling and interface details
- See Shehata et al. arxiv:2503.01787



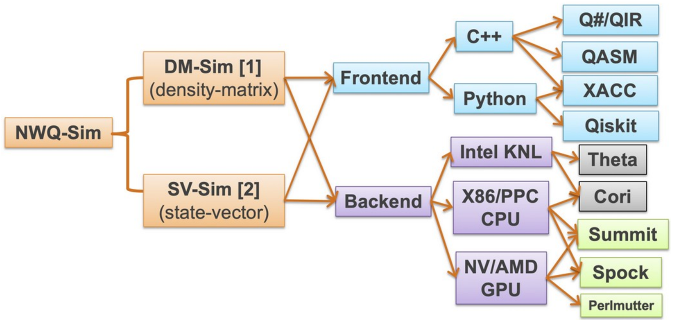
Amir Shehata

[Beck at al., Future Generation Computer Systems 161, 11-25 (2024)]

Simulators and Real Quantum Hardware

Simulators

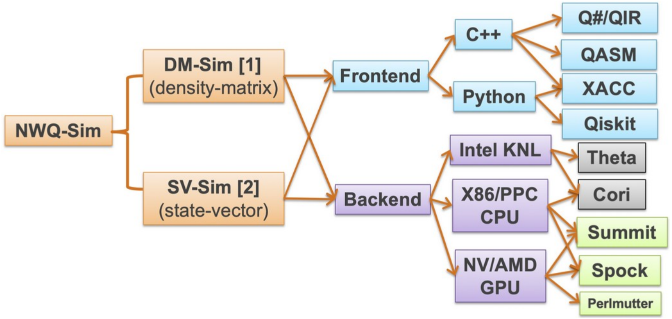
- Treat large-scale simulators as first-class citizens
  - Excellent for algorithm development
  - Noise studies



Simulators and Real Quantum Hardware

Simulators

- Treat large-scale simulators as first-class citizens
  - Excellent for algorithm development
  - Noise studies



Real Hardware

(potentially coming onsite later this year!)

- NV-center based hardware from Quantum Brilliance (QB)
  - More limited in quantum performance than other technologies
  - **…but good candidate for our integration effort**
  - QB installed a device into PAWSEY in Australia

NV-center based system

Outline

- Quantum Computing 101
- Integration of Quantum Hardware into HPC centers
- Benchmarking **joint** Quantum / HPC systems with **QStone**

- Benchmarking **joint** Quantum / HPC systems with **QStone**

Eduardo Antonio Coello Perez[1], Christopher Seck[1], David Rivas[2], In-Saeng Suh[1], Marco Ghibaudi[3]

1) OAK RIDGE National Laboratory

2) rigetti

3) river lane

Benchmarking

- Crucial in understanding performance metrics of complex systems
- What are the characteristics of good benchmarks?
  - Capture performance of some **relevant** part of the system
    - → Could be specific (e.g.: particular algorithm/application)
    - → Or general (e.g.: average performance of many algorithms)
  - Not easily “gamed”
  - Ideally agreed on/utilized by larger community

Benchmarking

- Crucial in understanding performance metrics of complex systems

- What are the characteristics of good benchmarks?
  - Capture performance of some **relevant** part of the system
    - → Could be specific (e.g.: particular algorithm/application)
    - → Or general (e.g.: average performance of many algorithms)
  - Not easily “gamed”
  - Ideally agreed on/utilized by larger community

- Actually hard to design “great” widely used benchmarks; in quantum many proposals, but not everyone agrees on details (no killer app **realized** yet?)

Benchmarking

- Crucial in understanding performance metrics of complex systems

- What are the characteristics of good benchmarks?
  - Capture performance of some **relevant** part of the system
    - → Could be specific (e.g.: particular algorithm/application)
    - → Or general (e.g.: average performance of many algorithms)
  - Not easily “gamed”
  - Ideally agreed on/utilized by larger community

- Actually hard to design “great” widely used benchmarks; in quantum many proposals, but not everyone agrees on details (no killer app **realized** yet?)

- **Our main goal**: to build a framework for **joint HPC-Quantum** benchmarking and **profiling** where it’s easy to utilize other already existing tools.

Benchmarking Quantum and Classical Systems

Very rich field: **MANY** benchmarks/metrics have been proposed

Classical Systems

HPC Challenge: HPL LINPACK, PTRANS, STREAM, etc…

Phoronix Test Suite

- Care about other metrics/parameters: FLOP count, Memory Size, Network speeds, etc..

Benchmarking Quantum and Classical Systems

Very rich field: **MANY** benchmarks/metrics have been proposed

Classical Systems

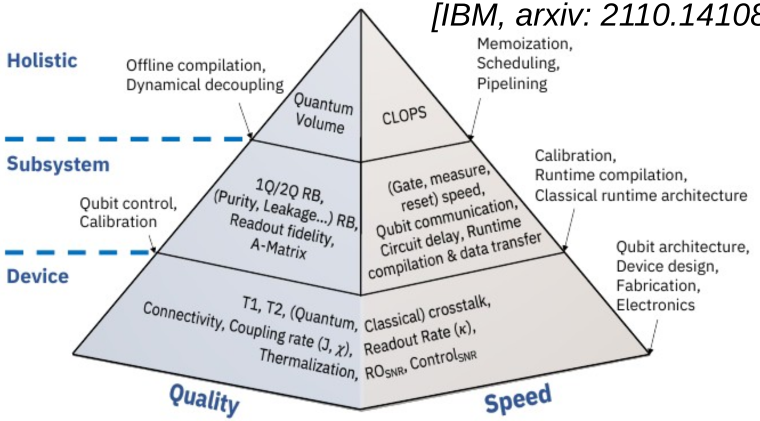
HPC Challenge: HPL LINPACK, PTRANS, STREAM, etc…

Phoronix Test Suite

- Care about other metrics/parameters: FLOP count, Memory Size, Network speeds, etc..

Quantum Systems

[IBM, arxiv: 2110.14108]



- Some metrics may be somewhat unique to quantum (e.g., coherence time, gate fidelity)

OAK RIDGE National Laboratory

Benchmarking Quantum and Classical Systems

Very rich field: **MANY** benchmarks/metrics have been proposed

Classical Systems

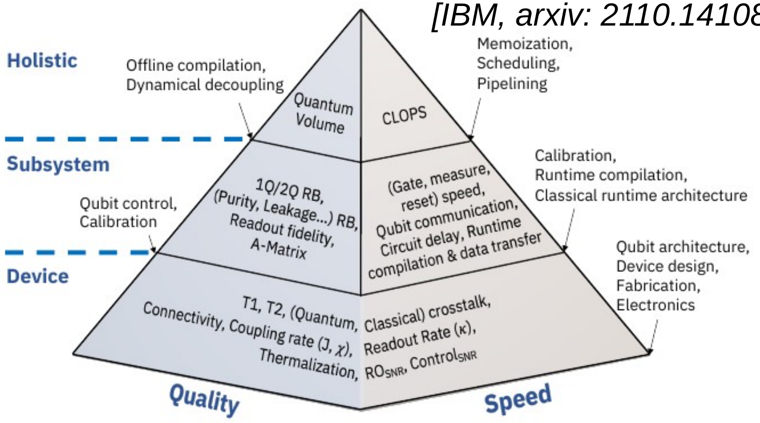
HPC Challenge: HPL LINPACK, PTRANS, STREAM, etc…

Phoronix Test Suite

- Care about other metrics/parameters: FLOP count, Memory Size, Network speeds, etc..

Quantum Systems

[IBM, arxiv: 2110.14108]



- Some metrics may be somewhat unique to quantum (e.g., coherence time, gate fidelity)

Various quantum benchmarking suites are available… but **limited tooling** that considers **hybrid HPC/quantum** performance

Oak Ridge National Laboratory

QStone: HPQ/QC Benchmarking Suite

- Joint effort with river lane & rigetti
- Free, **open-source** software framework for **combined** HPC/QC benchmarking

  https://github.com/riverlane/QStone

QStone

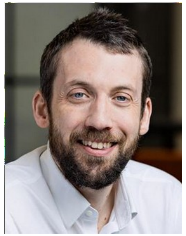
QStone: HPQ/QC Benchmarking Suite

- Joint effort with river lane & rigetti
- Free, **open-source** software framework for **combined** HPC/QC benchmarking

  https://github.com/riverlane/QStone

QStone

- Development lead by Riverlane



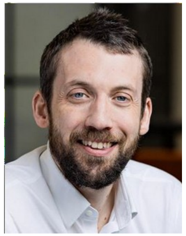
Marco Ghibaudi

QStone: HPQ/QC Benchmarking Suite

- Joint effort with riverlane & rigetti
- Free, **open-source** software framework for **combined** HPC/QC benchmarking

https://github.com/riverlane/QStone

QStone

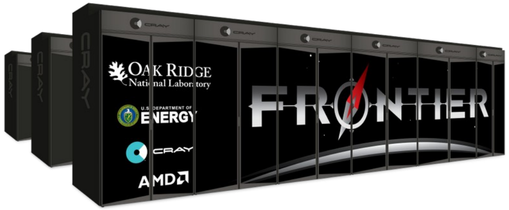


Marco Ghibaudi

- Development lead by Riverlane

summit

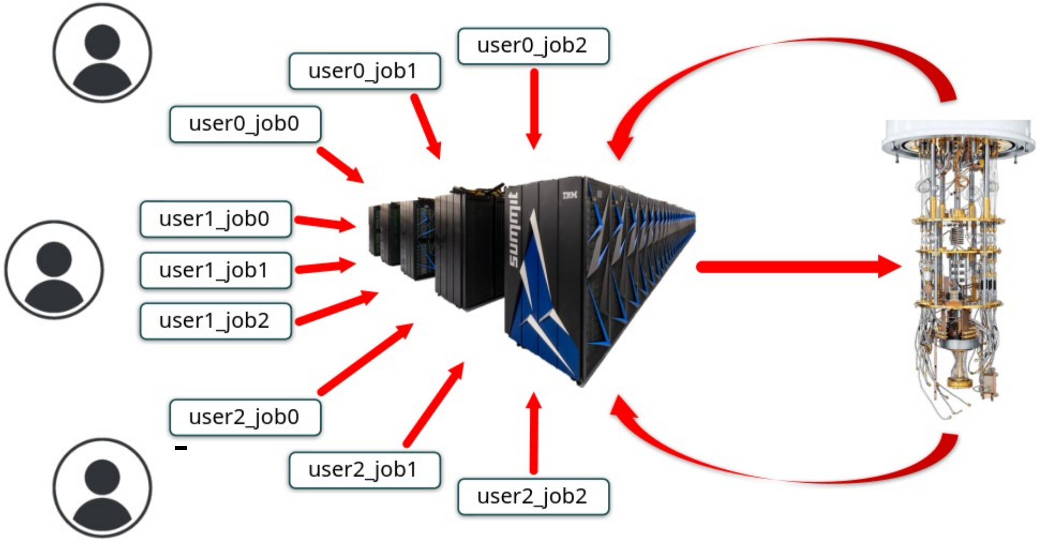
- Ongoing work includes development and testing:
  - With **offsite** rigetti hardware
  - Mock **onsite** hardware (riverlane control system)



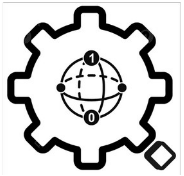
QStone: How Does It Work?

QStone

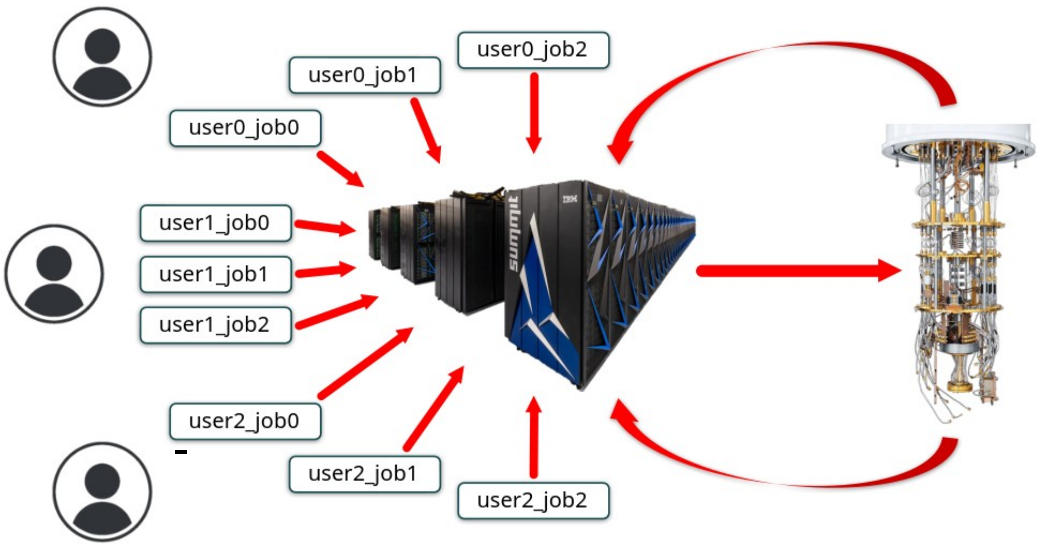
- Simple json-based “benchmarker”-provided configuration:
  (1) backend/connection details (type of connection, IP, etc.)
  (2) scheduler type to use (e.g. SLURM, CSM/jsrun, “bare metal”)
  (3) **number of “users” to simulate**
  (4) benchmarks (“apps”) that each user will run



QStone: How Does It Work?



- Simple json-based "benchmarker"-provided configuration:
  (1) backend/connection details (type of connection, IP, etc.)
  (2) scheduler type to use (e.g. SLURM, CSM/jsrun, "bare metal")
  (3) **number of "users" to simulate**
  (4) benchmarks ("apps") that each user will run



- QStone generates scheduler job configuration:

```
qstone generate -i conf.json
```

- Which can be ran on HPC (or other classical hardware):

```
qstone run -i scheduler.qstone.tar.gz
```

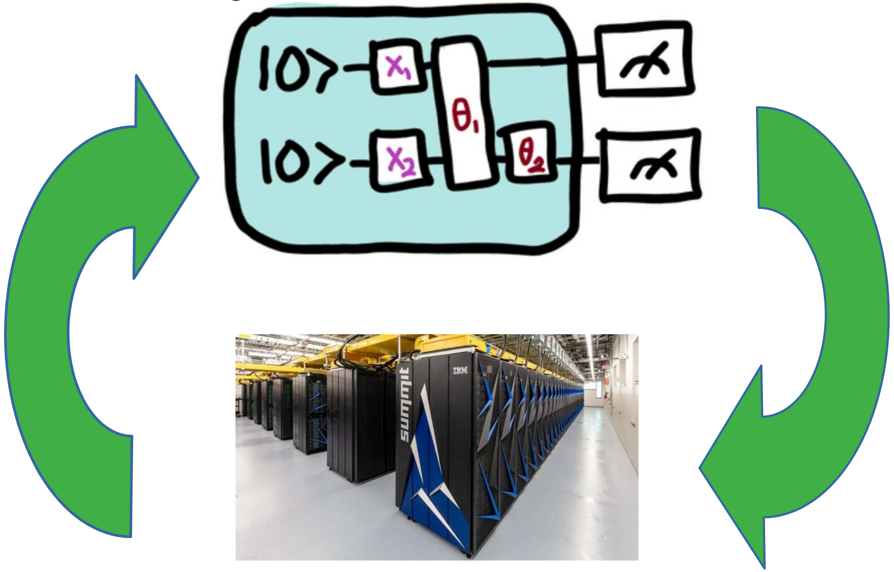
- Can do some data post-processing and organization (e.g.: build DataFrames)

QStone: What are the Benchmarks?

QStone

- Some benchmarking "apps" are built-in
  - QML-based variational classifier (optionally distributed)

Quantum execution



Classical optimization

[circuit image credit: Pennylane]

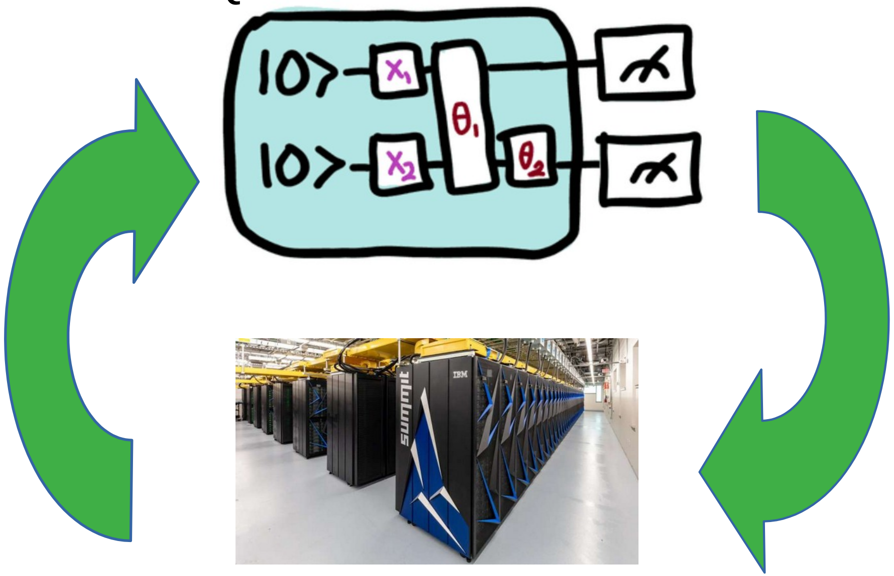
Oak Ridge National Laboratory

QStone: What are the Benchmarks?

QStone

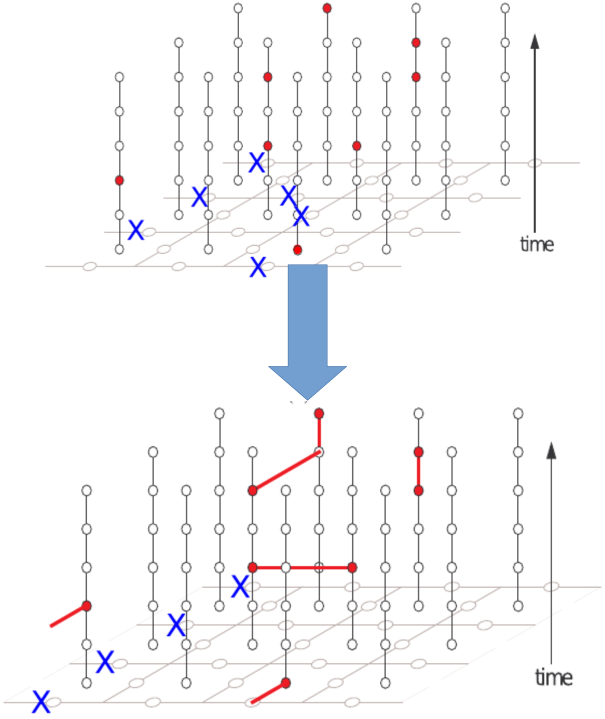
- Some benchmarking “apps” are built-in
  - QML-based variational classifier (optionally distributed)
  - QEC syndrome decoding (utilizing stim + pymatching)

Quantum execution



Classical optimization

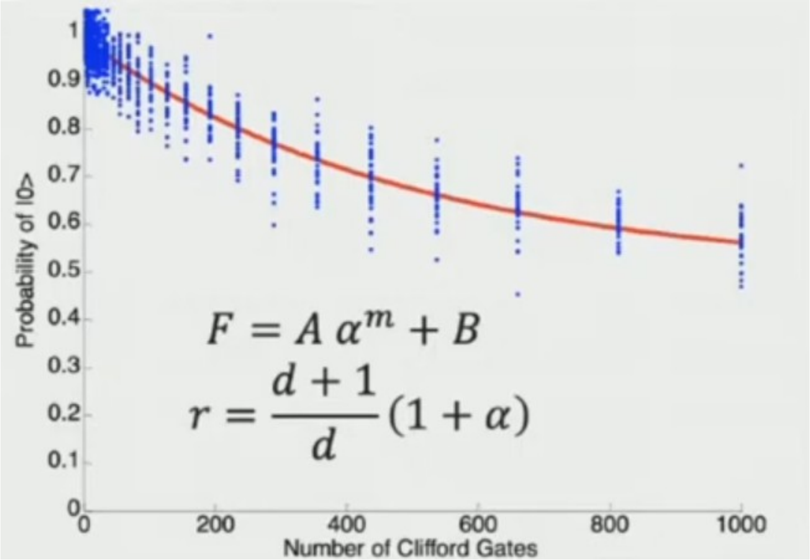
[circuit image credit: Pennylane]



QStone: What are the Benchmarks?

QStone

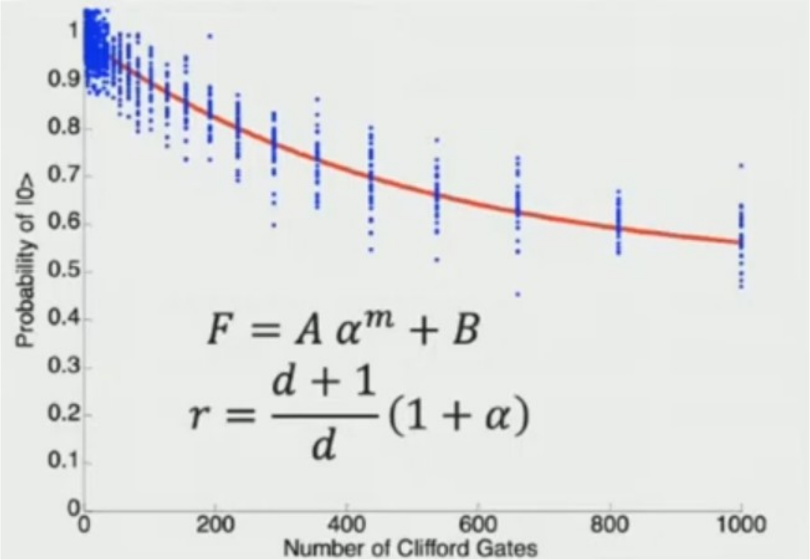
- Randomized benchmarking (utilizing pyGSTi)



QStone: What are the Benchmarks?

QStone

- Randomized benchmarking (utilizing pyGSTi)



- All these are user-configurable (eg.: number of qubits, shots, benchmark-specific params., etc)

- Currently exploring adding other "pre-defined" benchmarks, e.g.,
  - **Quantum Volume**
  - **VQE solver**

  …

- Ultimately goal is to add benchmarks that more fully utilize large scale HPC systems e.g:
  - KQD, SKQD [arXiv: 2405.05068, 2501.09702]
  - Op-Backpopagation [arXiv: 2502.01897]

Easily Expandable

QStone

- These may not be what users actually care about!

Easily Expandable

QStone

- These may not be what users actually care about!
- **Very easily user-expandable to arbitrary benchmarks**:
  - Need to define (some optional) steps:
  - **run:** core method that "runs stuff"
  - (optional) pre: e.g.: circuit preparation
  - (optional) post: e.g.: data postprocessing

```
from qstone.connectors import connector
from qstone.steps.computation import Computation

class NewType(Computation):

    def __init__(self):
    # Declare application variables from cfg file

    @trace(ComputationStep.PRE)
    def pre(self):
    # Preprocessing step

    @trace(ComputationStep.RUN)
    def run(self, connector.Connector):
    # Running step submitting quantum jobs

    @trace(ComputationStep.POST)
    def post(self):
    # Postprocessing step
```

Easily Expandable

QStone

- These may not be what users actually care about!

- **Very easily user-expandable to arbitrary benchmarks**:
  - Need to define (some optional) steps:
  - **run:** core method that “runs stuff”
  - (optional) pre: e.g.: circuit preparation
  - (optional) post: e.g.: data postprocessing

- Can measure timing of **arbitrary** operations (through python decorators)

```python
from qstone.connectors import connector
from qstone.steps.computation import Computation

class NewType(Computation):

    def __init__(self):
    # Declare application variables from cfg file

    @trace(ComputationStep.PRE)
    def pre(self):
    # Preprocessing step

    @trace(ComputationStep.RUN)
    def run(self, connector.Connector):
    # Running step submitting quantum jobs

    @trace(ComputationStep.POST)
    def post(self):
    # Postprocessing step
```

Easily Expandable

QStone

- These may not be what users actually care about!

- **Very easily user-expandable to arbitrary benchmarks**:
  - Need to define (some optional) steps:
  - **run:** core method that "runs stuff"
  - (optional) pre: e.g.: circuit preparation
  - (optional) post: e.g.: data postprocessing

- Can measure timing of **arbitrary** operations (through python decorators)

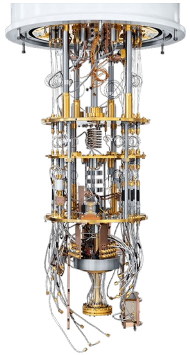
```
from qstone.connectors import connector
from qstone.steps.computation import Computation

class NewType(Computation):

    def __init__(self):
    # Declare application variables from cfg file

    @trace(ComputationStep.PRE)
    def pre(self):
    # Preprocessing step

    @trace(ComputationStep.RUN)
    def run(self, connector.Connector):
    # Running step submitting quantum jobs

    @trace(ComputationStep.POST)
    def post(self):
    # Postprocessing step
```

- **Thus "compatible" with other benchmarks already out there**

Testing Setup

Two “quantum devices”:

Offsite: real Rigetti hardware in California



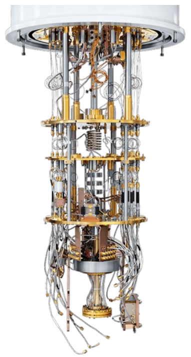
(ongoing effort)

- Subject to “double queue problem”
- Reservations may not coincide with when HPC chooses to run jobs
- Less control over QC side → highlights complexities of offsite hardware

Testing Setup

Two “quantum devices”:

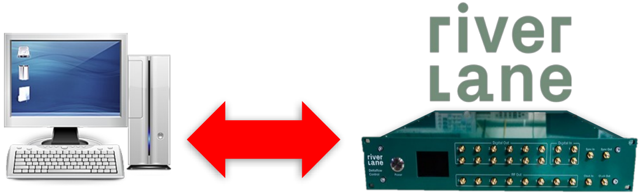
Offsite: real Rigetti hardware in California



- Subject to “double queue problem”
- Reservations may not coincide with when HPC chooses to run jobs
- Less control over QC side → highlights complexities of offsite hardware

(ongoing effort)

Onsite: emulated mock device

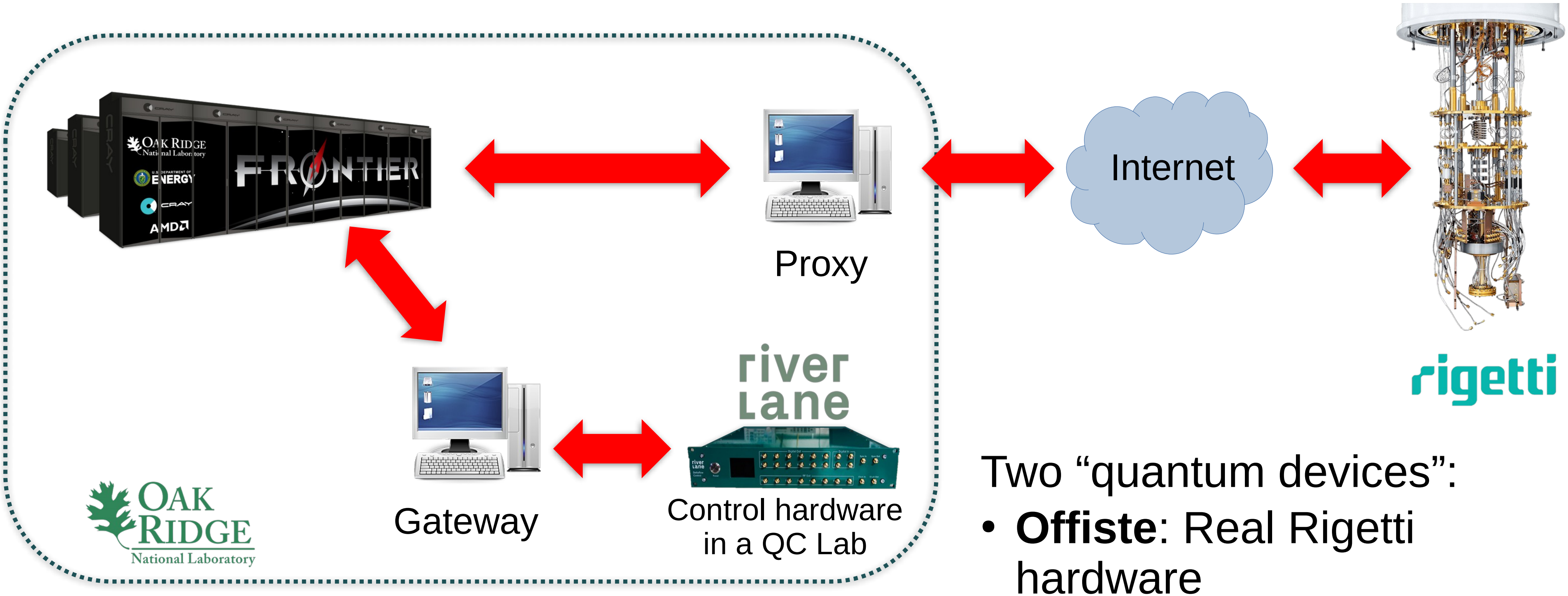


Gateway

Control hardware in a QC Lab

- Gateway runs a server that dispatches instructions to control hardware
- Control hardware simulates actual pulse timing for a trapped ion or SC device
- Excellent for exploration/testing; have full control over all systems

Testing Setup: Full Story



Two “quantum devices”:

- **Offiste**: Real Rigetti hardware
- **Onsite**: Emulated mock device (simulates pulse timings)

(Rigetti integration is ongoing)

Resulting Data

[simplified configuration]

```
{
    "project_name" : "CSC562",
    "connector" : "HTTPS",
    "qpu_ip_address" : "http://10.1.247.27",
    "qpu_port" : "8090",
    "num_required_qubits" : 16,
    "qpu_management" : "",
    "users" : [
        {
        "user": "user0",
        "weight" : 1.0,
        "computations" :
            {
            "type0": 0.33,
            "type1": 0.33,
            "type2": 0.33
            }
        },
        {
        "user": "user1",
        "weight" : 1.0,
        "computations" :
            {
            "type0": 0.1,
            "type1": 0.1,
            "type2": 0.8
            }
        },
```

[...]

	user	prog_id	job_id	job_type	job_step	label	start	end	total
0	user0	0	0	CONNECTION	POST	BITSTRING_COUNTING	1128769272092506	1128769274528999	2436493
0	user0	0	0	CONNECTION	RUN	CIRCUIT_RUNNING	1128770128262694	1128770132090455	3827761
0	user0	0	0	TYPE4	RUN	QASM_GENERATION	1128769942763639	1128769942829754	66115
0	user0	0	0	TYPE4	RUN	QASM_GENERATION	1128769171101246	1128769171165539	64293
0	user0	0	0	CONNECTION	RUN	CIRCUIT_RUNNING	1128769215976829	1128769219301374	3324545
...	...	...	...	...	...	...	...	...	...
0	user0	0	0	CONNECTION	PRE	QASM_READING	1128769253927671	1128769254072536	144865
0	user0	0	0	TYPE4	RUN	QASM_GENERATION	1128770127905550	1128770127966693	61143
0	user0	0	0	TYPE4	RUN	QASM_GENERATION	1128770774390610	1128770774456652	66042
0	user0	0	0	CONNECTION	PRE	QASM_READING	1128769278131578	1128769278268935	137357
0	user0	0	0	TYPE4	RUN	QASM_GENERATION	1128769471463218	1128769471528500	65282

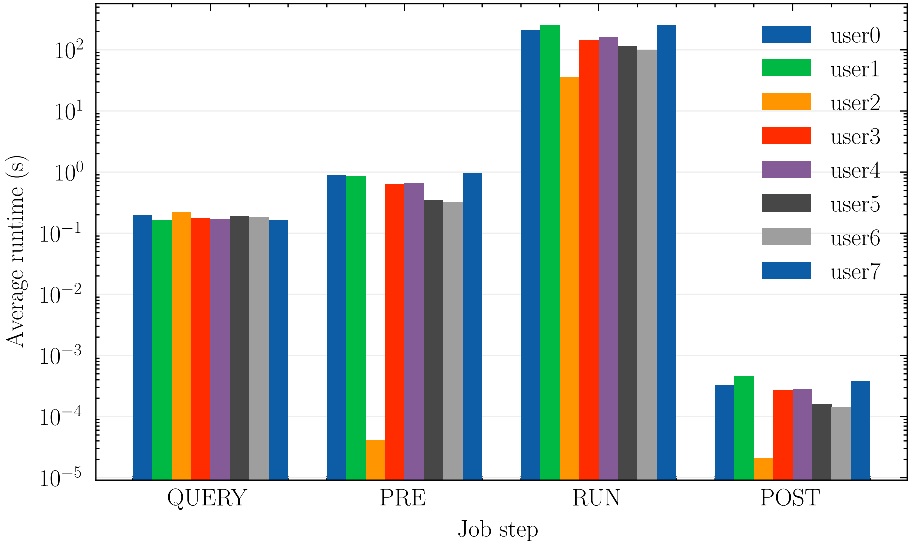
```
qstone generate -i conf.json
```

```
qstone run -i scheduler.qstone.tar.gz
```

```
qstone profile --cfg conf.json --folder qstone_profile
```

QStone: Example Results (Science Perspective)

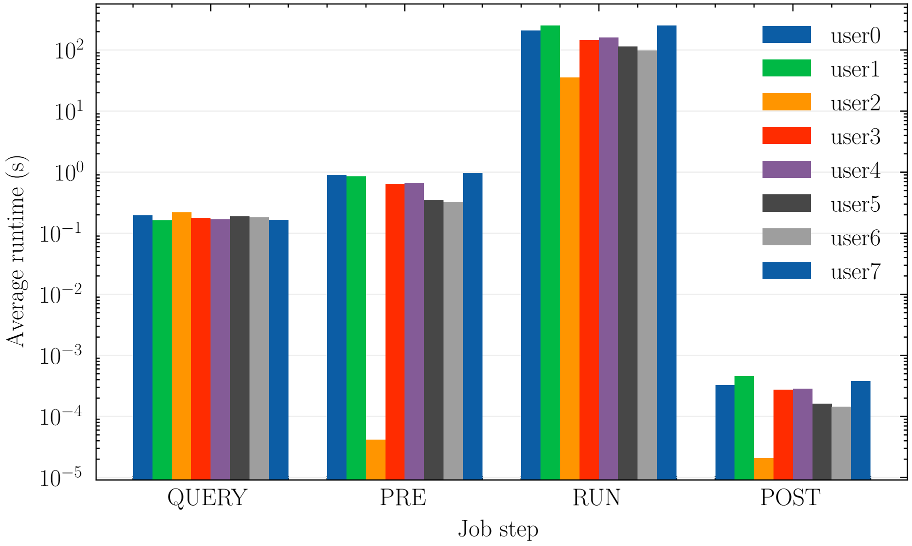
Can help scientists understand application characteristics/performance

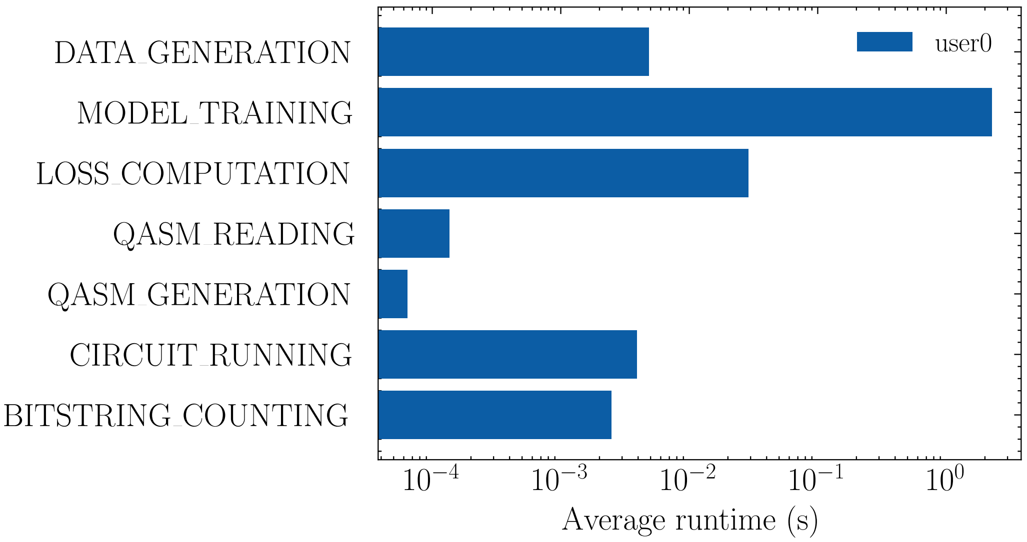


How much each user spends (on average)
in different computation steps

QStone: Example Results (Science Perspective)

Can help scientists understand application characteristics/performance





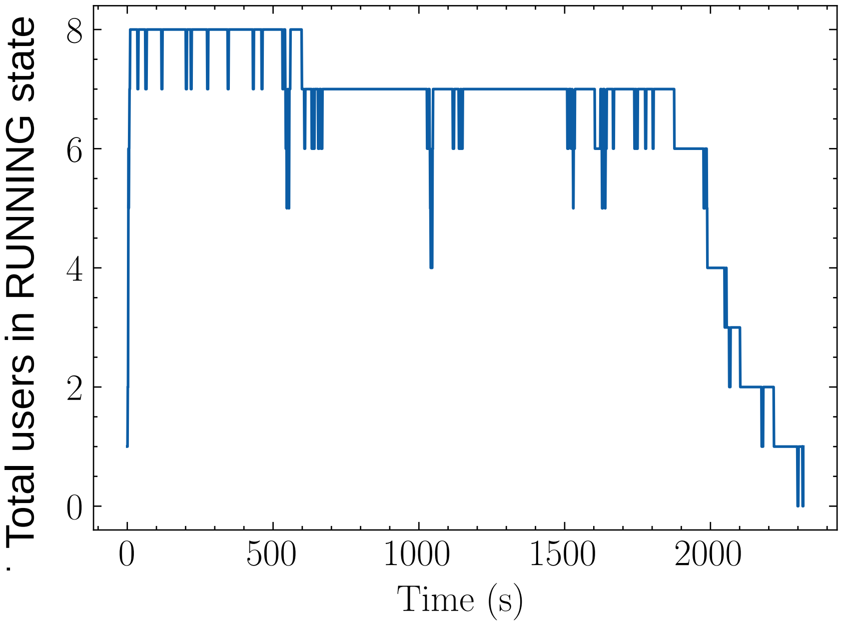
How much each user spends (on average) in different computation steps

How costly are different components of application logic (here for QML classifier “app”)

(toy example, local development run)

QStone: Example Results (System Performance)

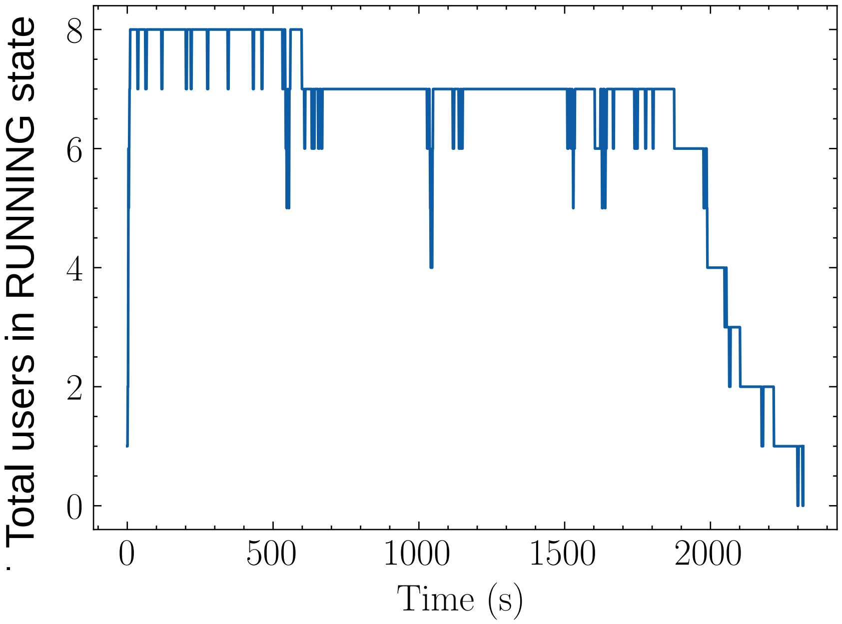
Can help (HPC) system engineers optimize e.g. scheduler performance

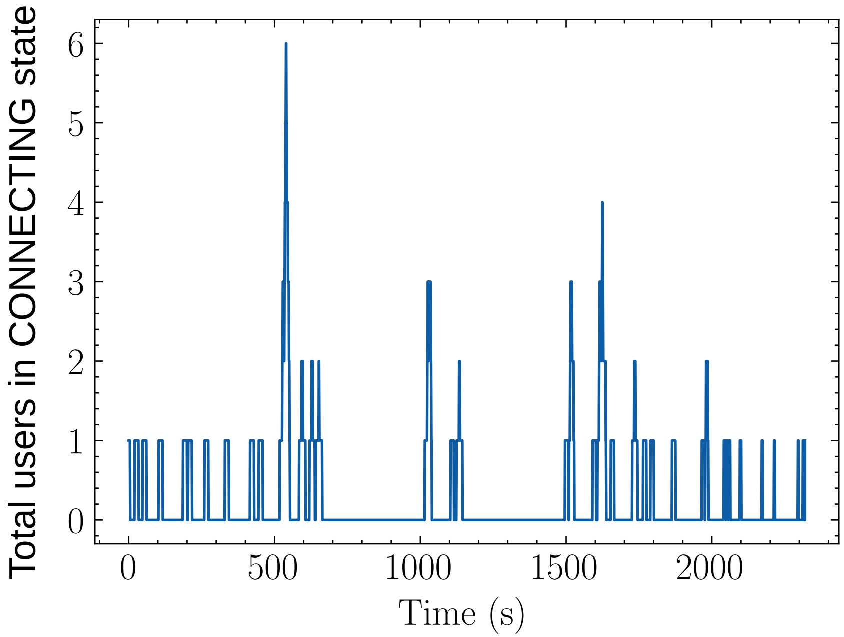


How many users are RUNNING computations at any given time?

QStone: Example Results (System Performance)

Can help (HPC) system engineers optimize e.g. scheduler performance



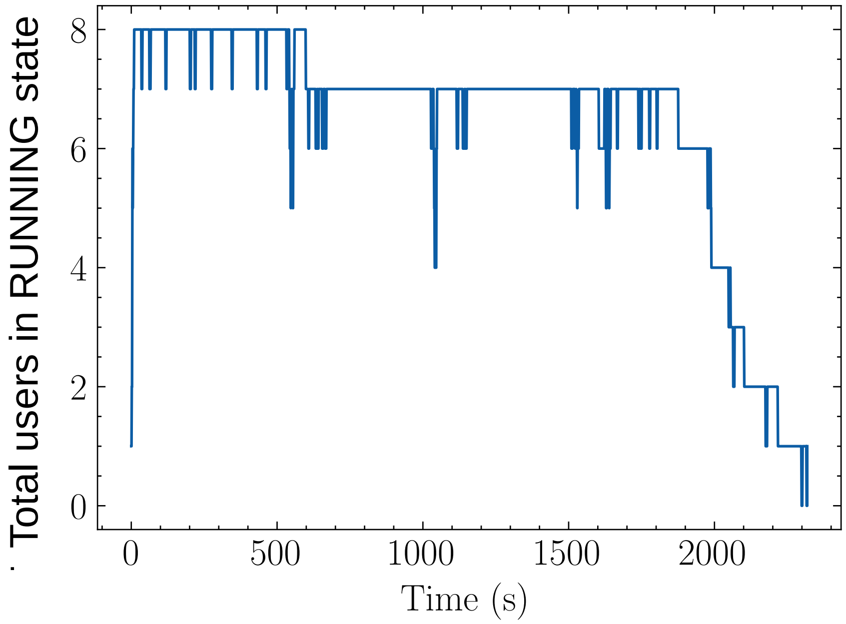


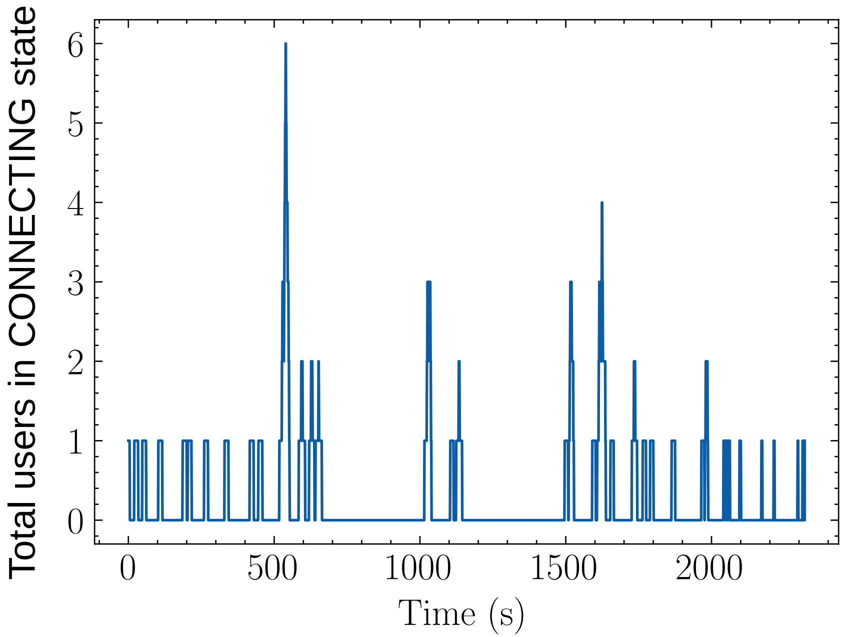
How many users are RUNNING computations at any given time?

How many users are CONNECTING to QPUs at any given time?

QStone: Example Results (System Performance)

Can help (HPC) system engineers optimize e.g. scheduler performance





How many users are RUNNING computations at any given time?

How many users are CONNECTING to QPUs at any given time?

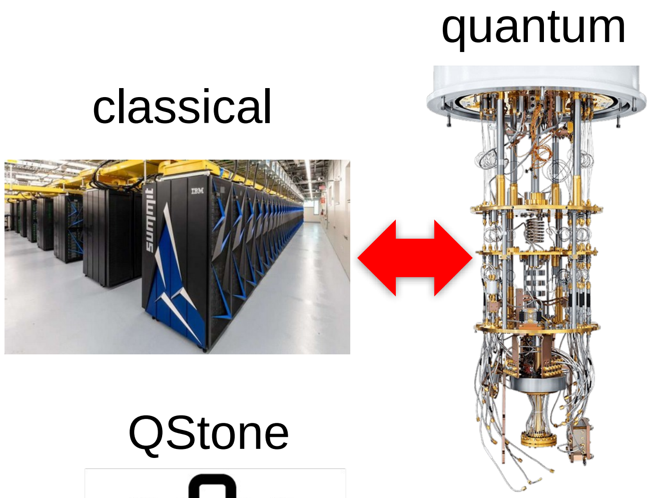
Can keep track of ALL-THE-THINGS (but can add more, e.g., power consumption)
...can study patterns, correlations, etc.

Conclusions

- Integration of Quantum Hardware into HPCs ecosystems:
  - ORNL building a “middleware” framework
    - *Beck et al. Fut. Gen. Com. Sys. 161, 11-25 (2024)*
    - *Shehata, et al, arXiv:2503.01787 (2025)*

- **QStone**: **free**, **open-source** python-based benchmarking suite for **joint HPC-Quantum workflows**
  - Easy to use
  - Flexible and easily expandable

Preprint coming soon!



QStone

https://github.com/riverlane/QStone

OAK RIDGE National Laboratory

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Extra slides

ORNL Classical Computing: **INCITE** and **DDs**

Computational resources allocations through Oak Ridge Leadership Computing Facility (**OLCF**)

Innovative and Novel Computational Impact on Theory and Experiment (**INCITE**) Program

- For **high-impact**, **computationally intensive** research campaigns in a broad array of science, engineering and computer science domains
- Very competitive
- Requires already written software stack (GPU-ready, parallelized, etc)
- Call for proposals: once a year

Director's Discretionary (**DD**) Program

- For smaller computational studies
  - (e.g., ~20-50k node hours)
- Good for scientific campaigns and **tools development** to get ready for INCITE
- Call for proposals: any time(!)
- ORNL liaisons (that can help!)

Historically not "too many" quantum-computing related proposals

Info: https://www.olcf.ornl.gov/

Contact: groszkowskip@ornl.gov

Oak Ridge National Laboratory