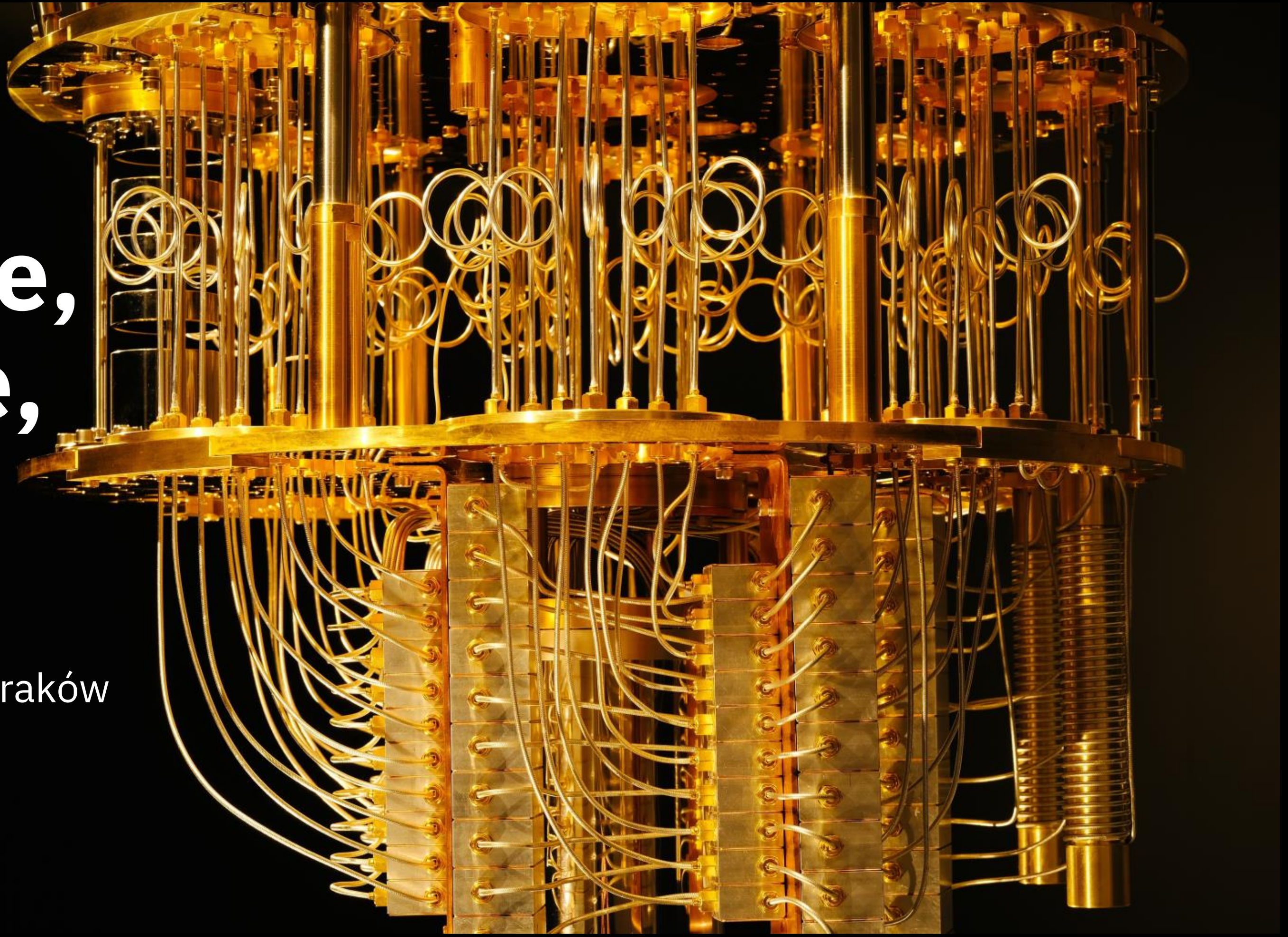
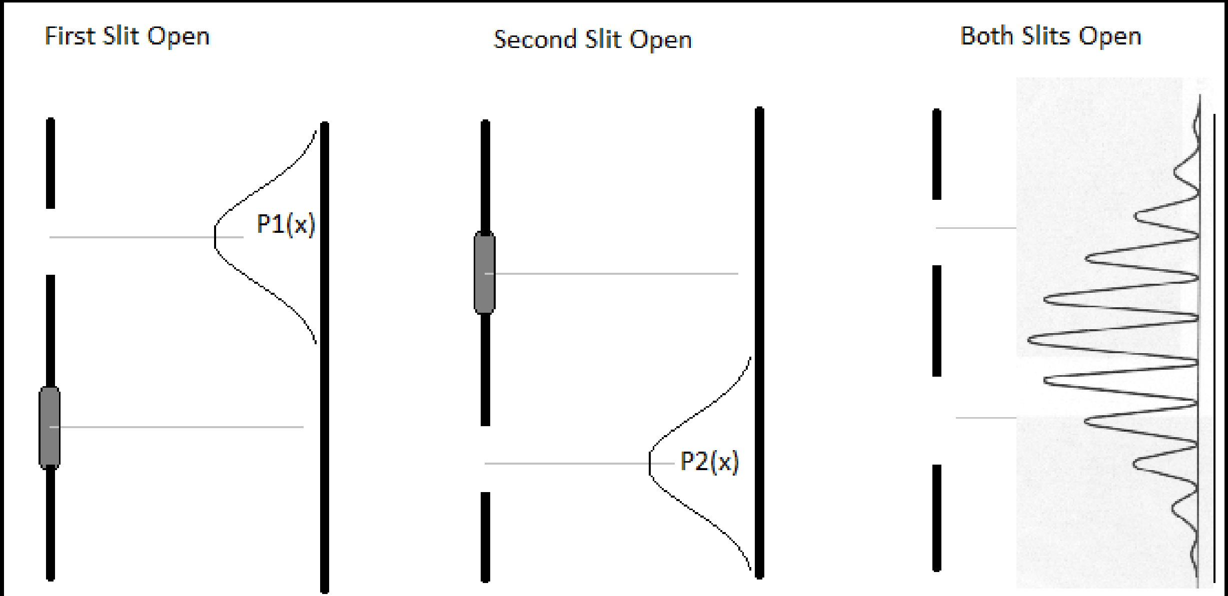
IBM Q: hardware, software, future...



Tomasz Stopa
IBM Software Laboratory, Kraków

IBM

Quantum nature of matter



Qubit

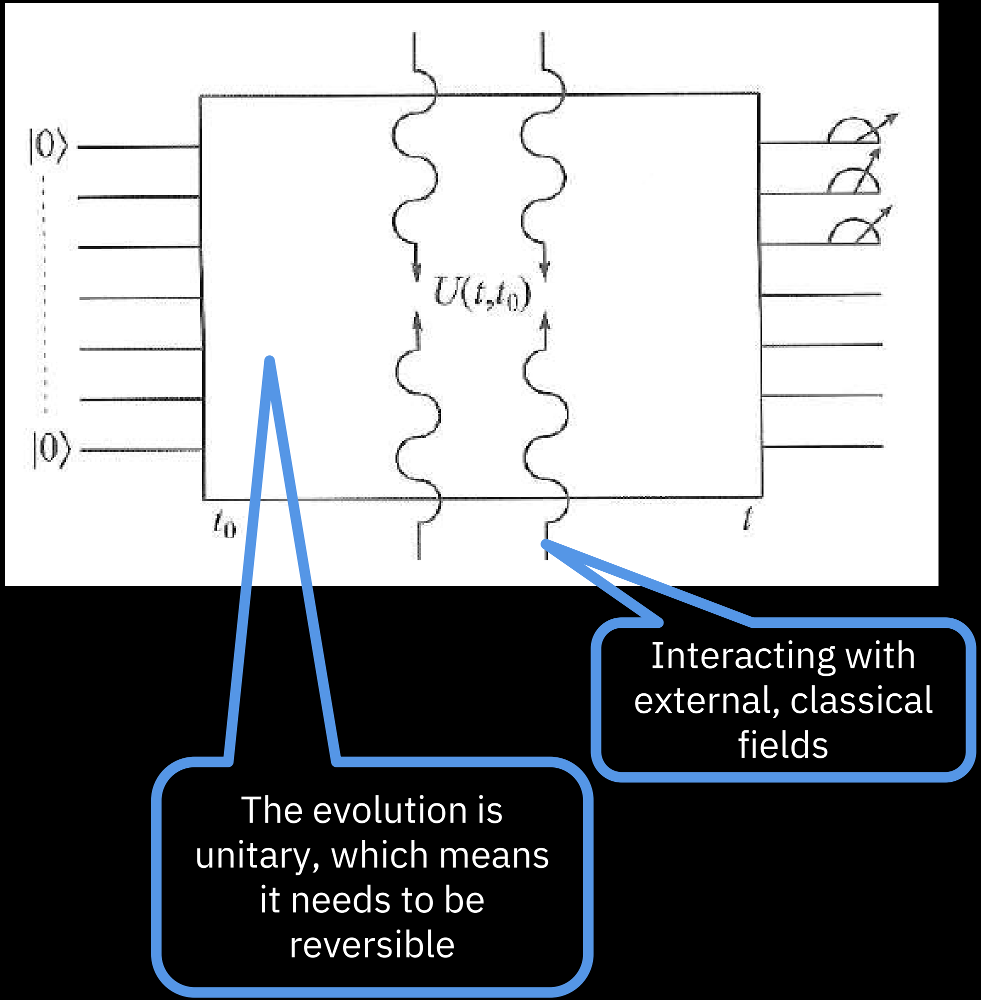
- Two state system: |0> and |1>
- We are able to stear it defining it’s state and evolution
- Qubit state needs to be constant in time (if not modified intentionally)
- Qubit can be in a superposition of both states at the same time

General concept of Quantum Computer

1. We prepare n qubits in states |0 >

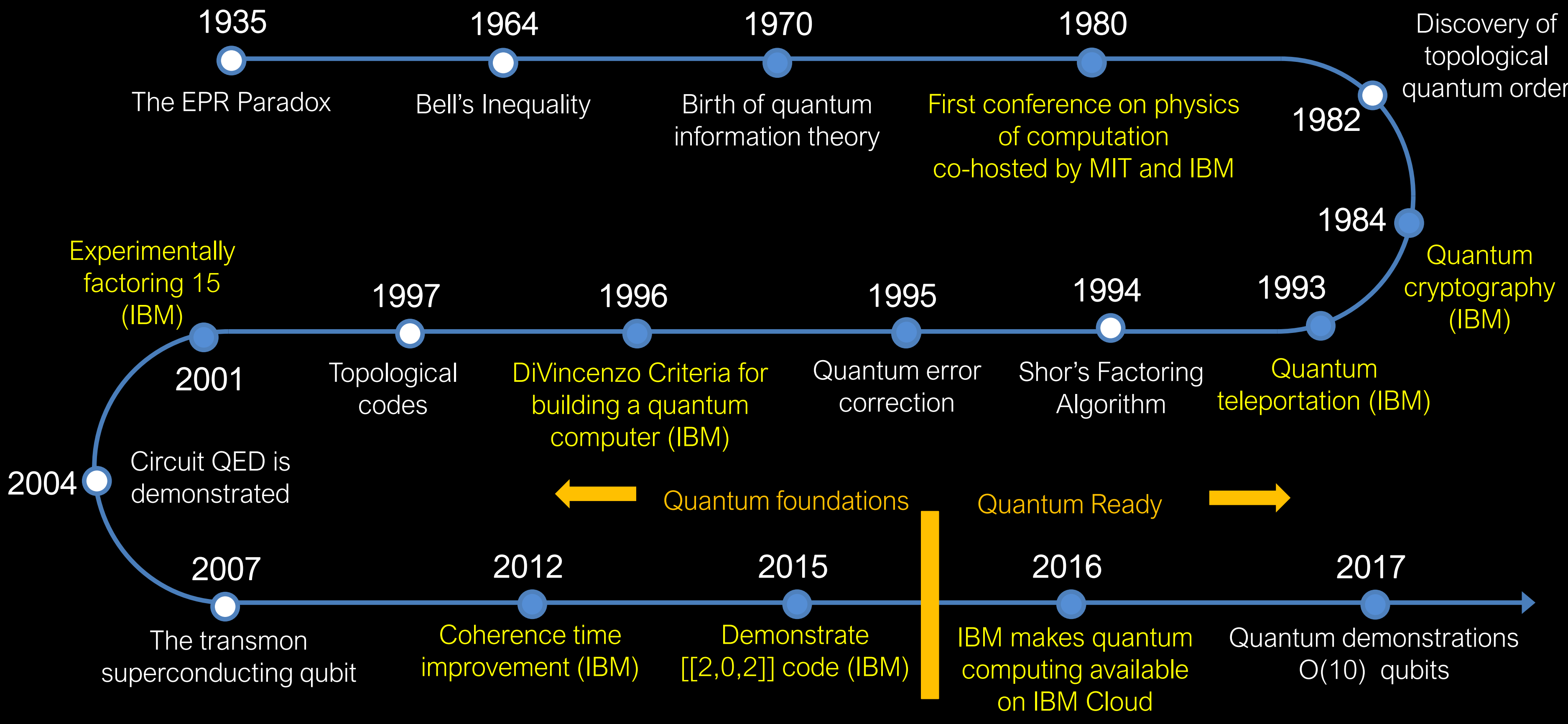
2. Then the system deterministically evolves. We stear the qubits with external fields to implement the necessary operations on the qubits

3. We measure all the qubits or their subset (3 on the picture above)



Significant Events in Quantum Computing

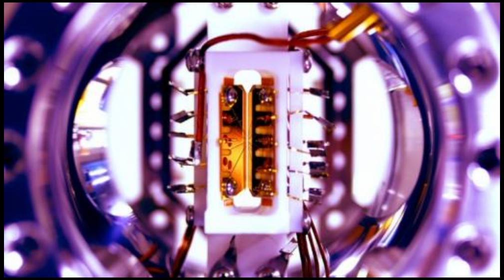
IBM Q



hardware...

Quantum Computing Implementation Technologies

Ions



Credit: S. Debnath and E. Edwards/JQI
Monroe Group, University of Maryland/JQI

Photons

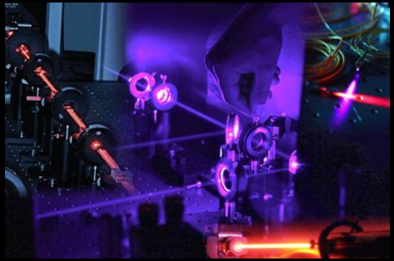
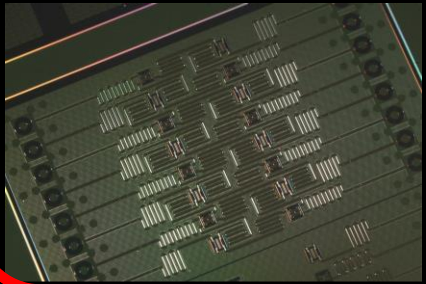
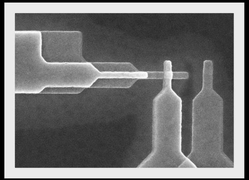


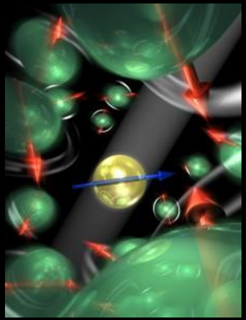
Image from the Centre for Quantum Computation & Communication Technology, credit Matthew Broome

Superconducting Circuits





Solid-state defects



NV Centers, Phosphorous in Si, SiC defects, etc.

Image from Hanson Group, Delft

Controllability

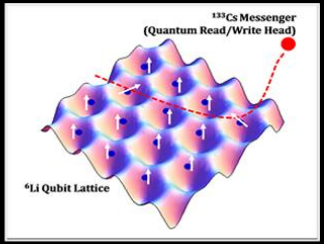
Balancing trade-space to maximize quantum volume

Coherence

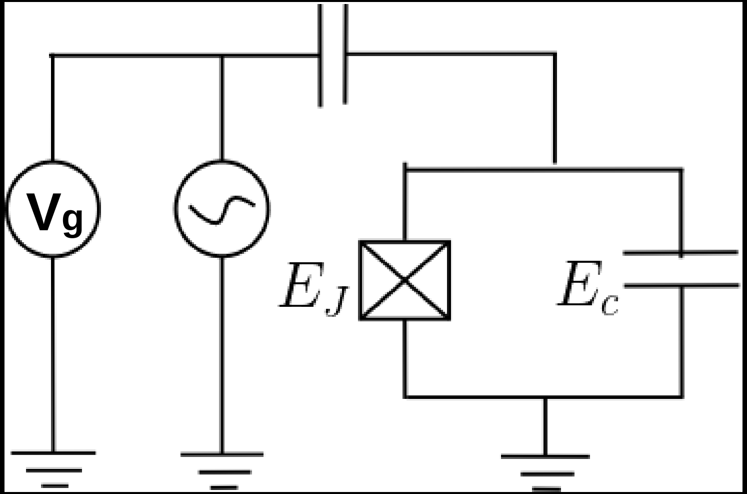
Connectivity

Neutral Atoms

Image from Cheng Group, University of Chicago



Superconducting Transmon Qubits

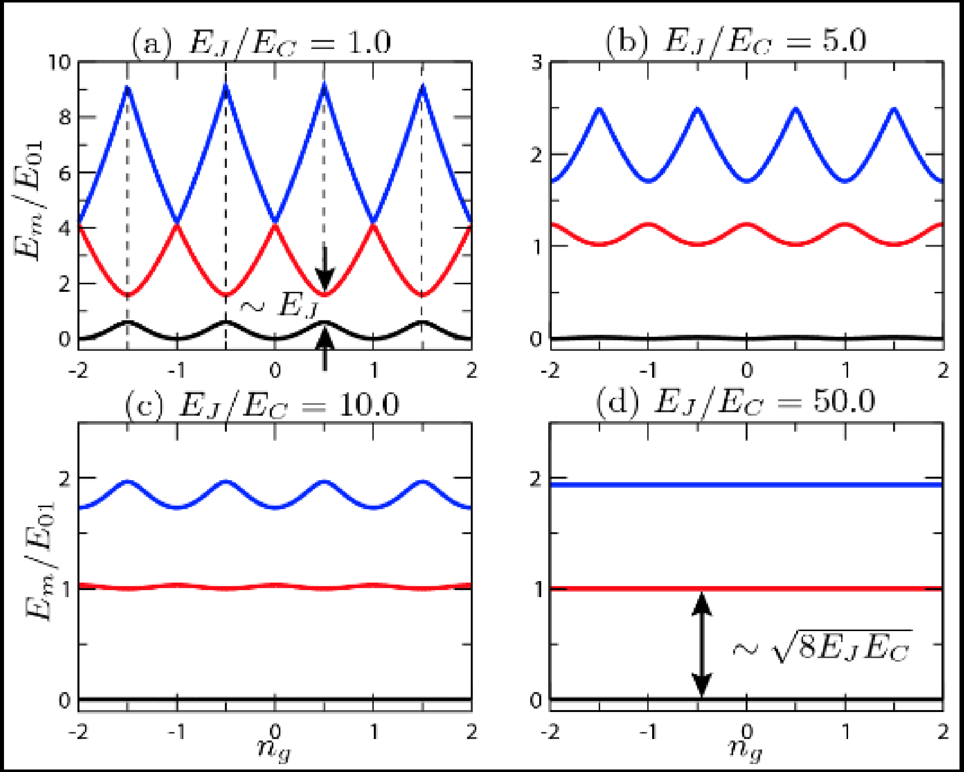


$$\hat{H} = 4E_c(\hat{n} - n_g)^2 - E_J \cos(\hat{\phi})$$

- V_g – external noise; always present but we want to minimize it’s impact
- JJ – needed instead of inductance to make the system anharmonic

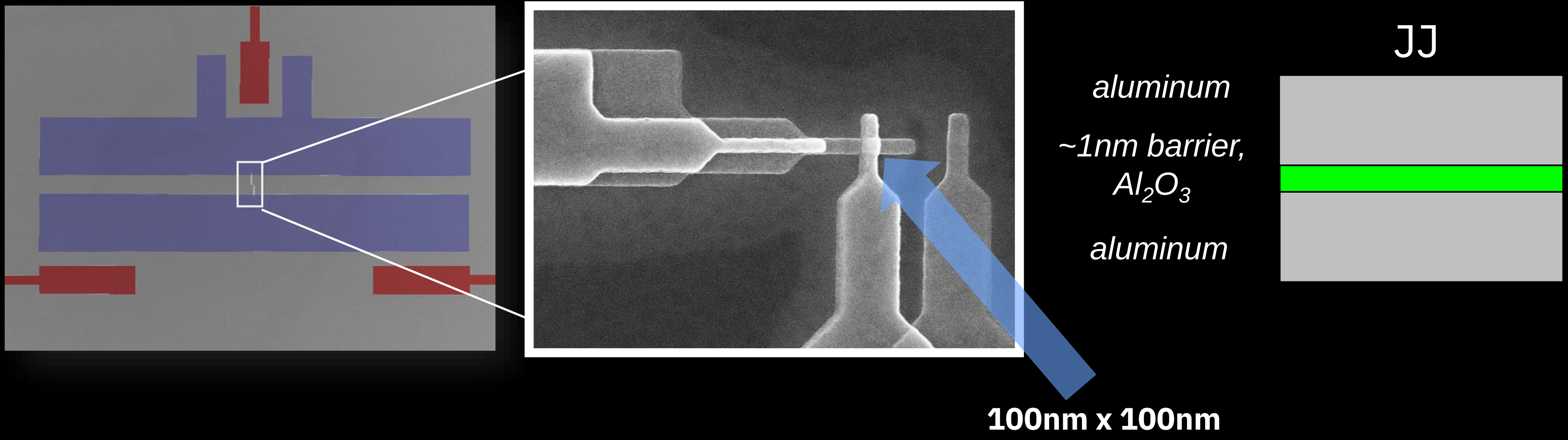
Transmon pioneered by Schoelkopf group, from Yale University
Koch et. al. PRA **76**, 04319 (2007)

Superconducting Transmon Qubits – cont.



- $E_J >> E_C$ ➔ transmon qubit (E_J/E_C~50)
- E_{01} independent from n_g
- $E_J \approx$20 GHz, Ec $\approx$ 400 MHz
- $E_{12} < E_{01}$ ➔ we are still good ;)
- $E_{01} \approx$ 5 GHz $\approx$ 240 mK

Transmon Qubit Parameters

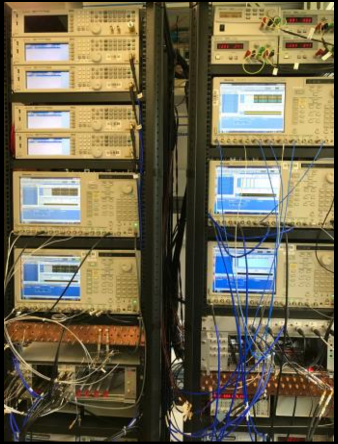


- JJ is very small comparing to capacitor
- Size of the qubit is ~300-500 μm

IBM Q quantum computing systems

Room Temperature

Cosmic Microwave Background 2.7K



Microwave electronics

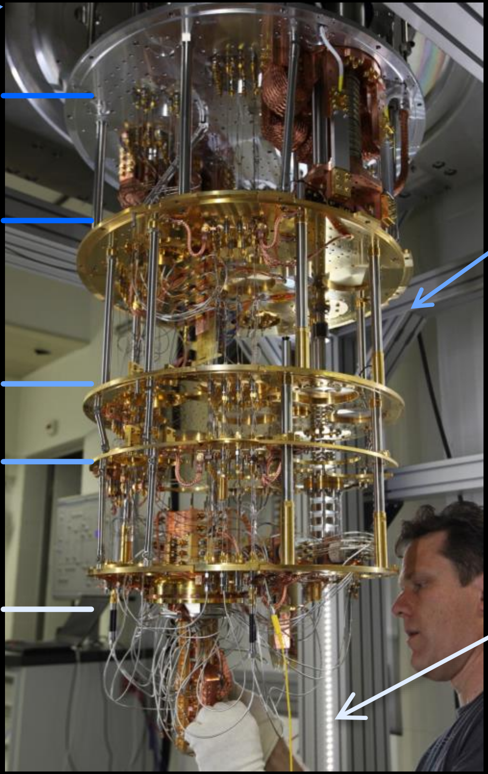
40K

3K

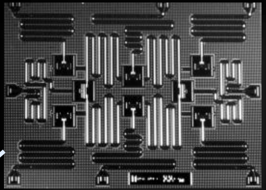
0.9K

0.1K

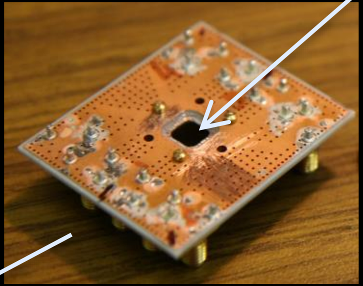
0.015K



Refrigerator to cool qubits to 15 mK with a mixture of ^{3}He and ^{4}He



Chip with superconducting qubits and resonators



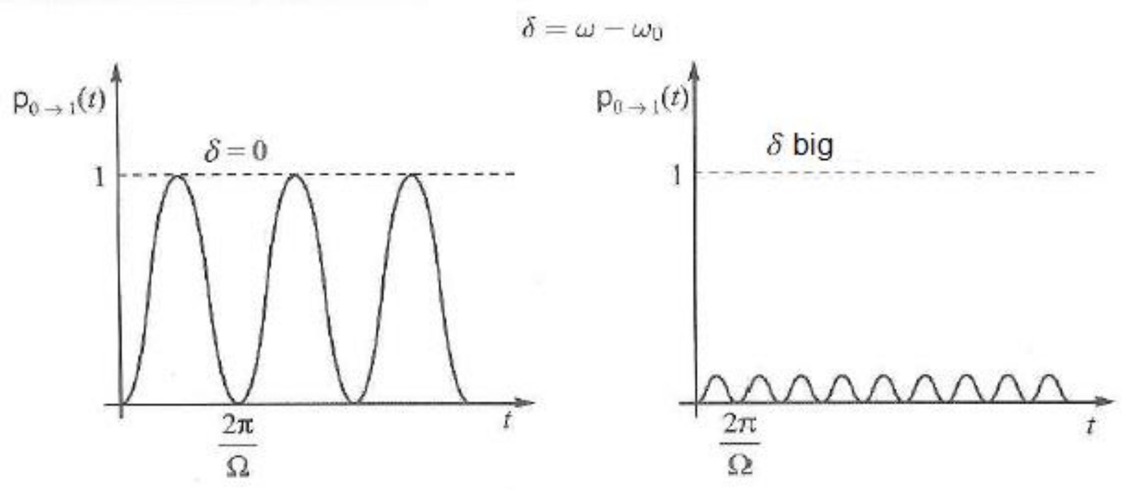
PCB with the qubit chip at 15 mK
Protected from the environment by multiple shields

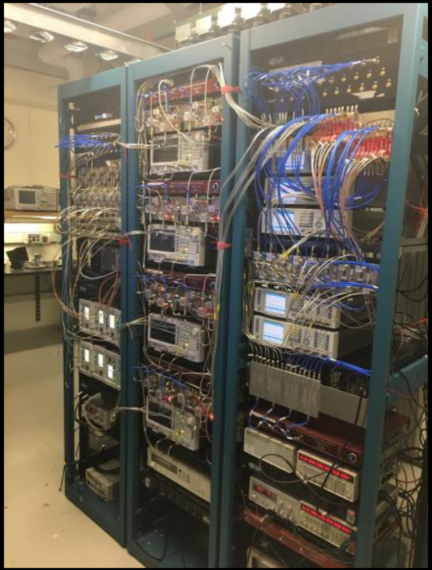
Changing qubit state with Rabi oscillations

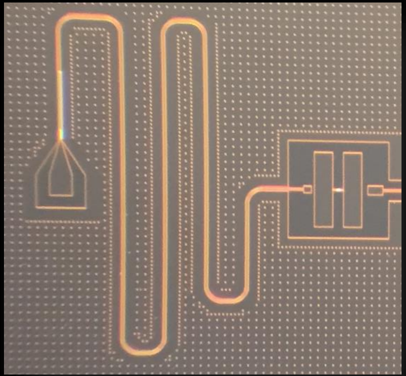
- Operations on qubits are based on Rabi oscillations induces with **microwave pulses**

- Rabi oscillations appear when we apply periodically changing external electric or magnetic fields to qubits

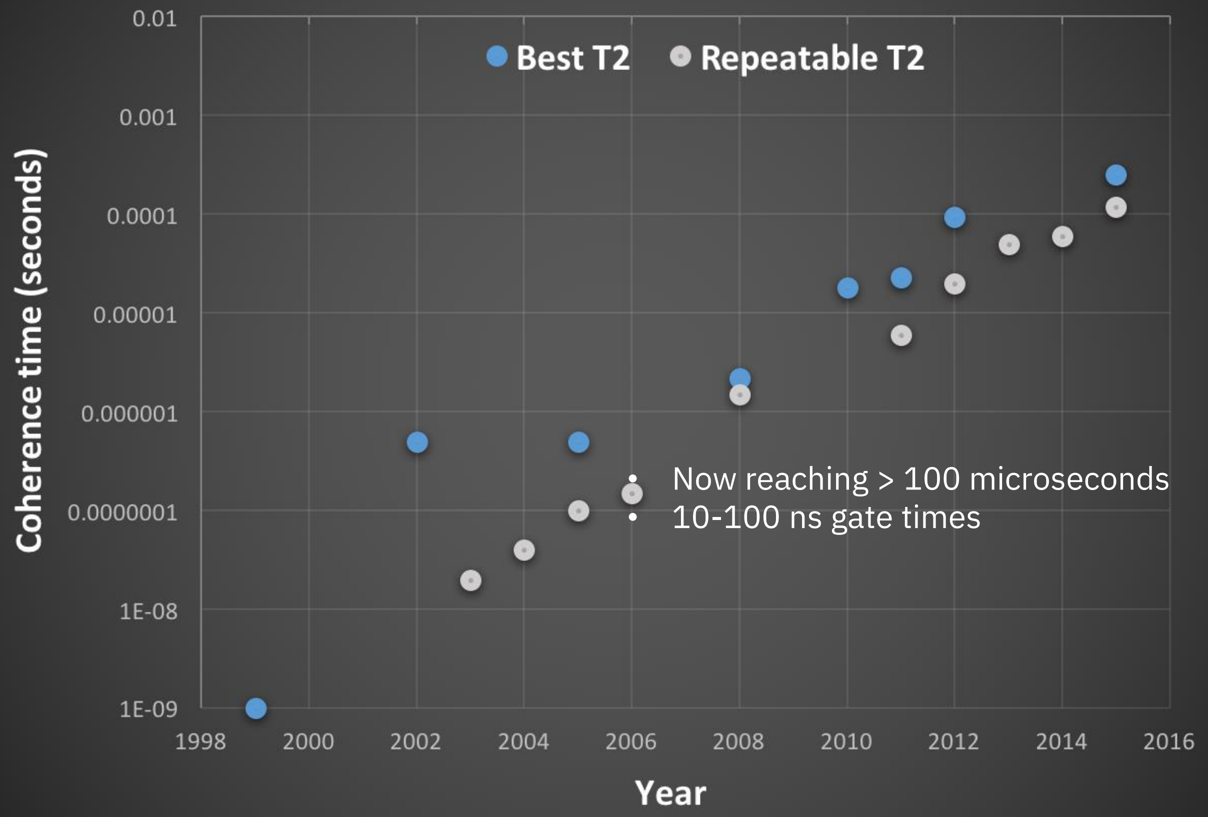
- To change |0 > into |1 > one needs to apply external field of exactly calculated frequency for precisely calculated time T







Coherence times of superconducting qubits



- Now reaching > 100 microseconds
- 10-100 ns gate times

- Developments to extend coherence times
  - Materials e.g. [2]
  - Design and geometries e.g. [3]
  - 3D transmon [4]
  - IR Shielding [5,6],
  - Cold normal metal cavities and cold qubits [7]
  - High Q cavities [8]
  - Titanium Nitride (collaboration with David Pappas @ NIST Boulder) [9] ...
- Remarkable progress over the past decade

[2] J. Martinis *et al.*, PRL **95** 210503 (2005)
[3] K. Geerlings *et al.*, APL 192601 (2012)
[4] H. Paik *et al.*, PRL **107**, 240501 (2011)
[5] R. Barends *et al.*, APL **99**, 113507 (2011)
[6] A. Corcoles *et al.*, APL **99**, 181906 (2011)
[7] C. Rigetti *et al.*, PRB **86**, 100506 (2012)
[8] M. Reagor *et al.*, arXiv:1302.4408 (2013)
[9] J. Chang et al. APL 103, 012602 (2013)

IBM-Q → devices currently available

Client devices

20 qubits

- IBM Q 20 Tokyo

Public devices

14 qubits

- IBM Q 14 Melbourne

16 qubits

- IBM Q 16 Rüschlikon

5 qubits

- IBM Q 5 Tenerife

Simulators

32 qubits

- IBM Q QASM 32 Q Simulator

Retired devices

20 qubits

- IBM Q 20 Austin

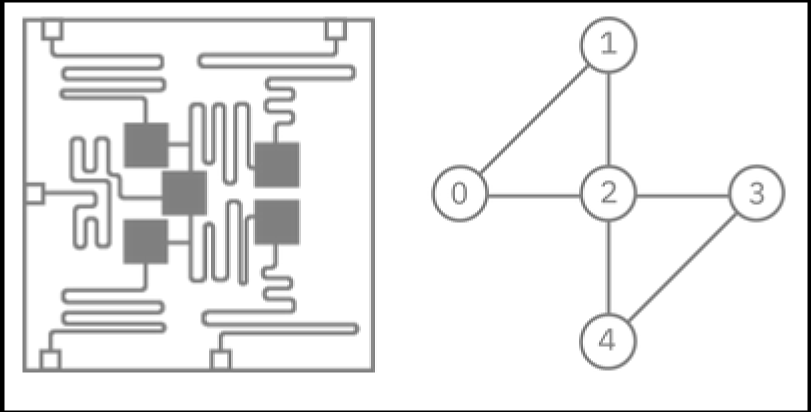
5 qubits

- IBM Q 5 Yorktown

IBM Q 5 Tenerife

5

qubits



Average measurements

Frequency (GHz)	5.25
T1 (μs)	47.50
T2 (μs)	42.10
Gate error (10^{-3})	0.69
Readout error (10^{-2})	7.90

IBM Q 14 Melbourne

14

qubits

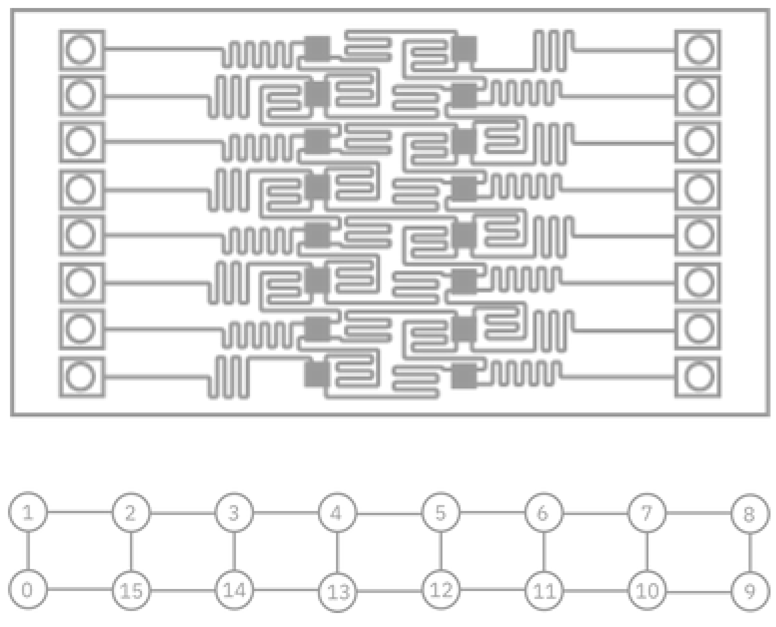
Average measurements

Frequency (GHz)	5.10
T1 (μs)	59.10
T2 (μs)	15.70
Gate error (10^{-3})	2.11
Readout error (10^{-2})	3.47

IBM Q 16 Rüschlikon

16

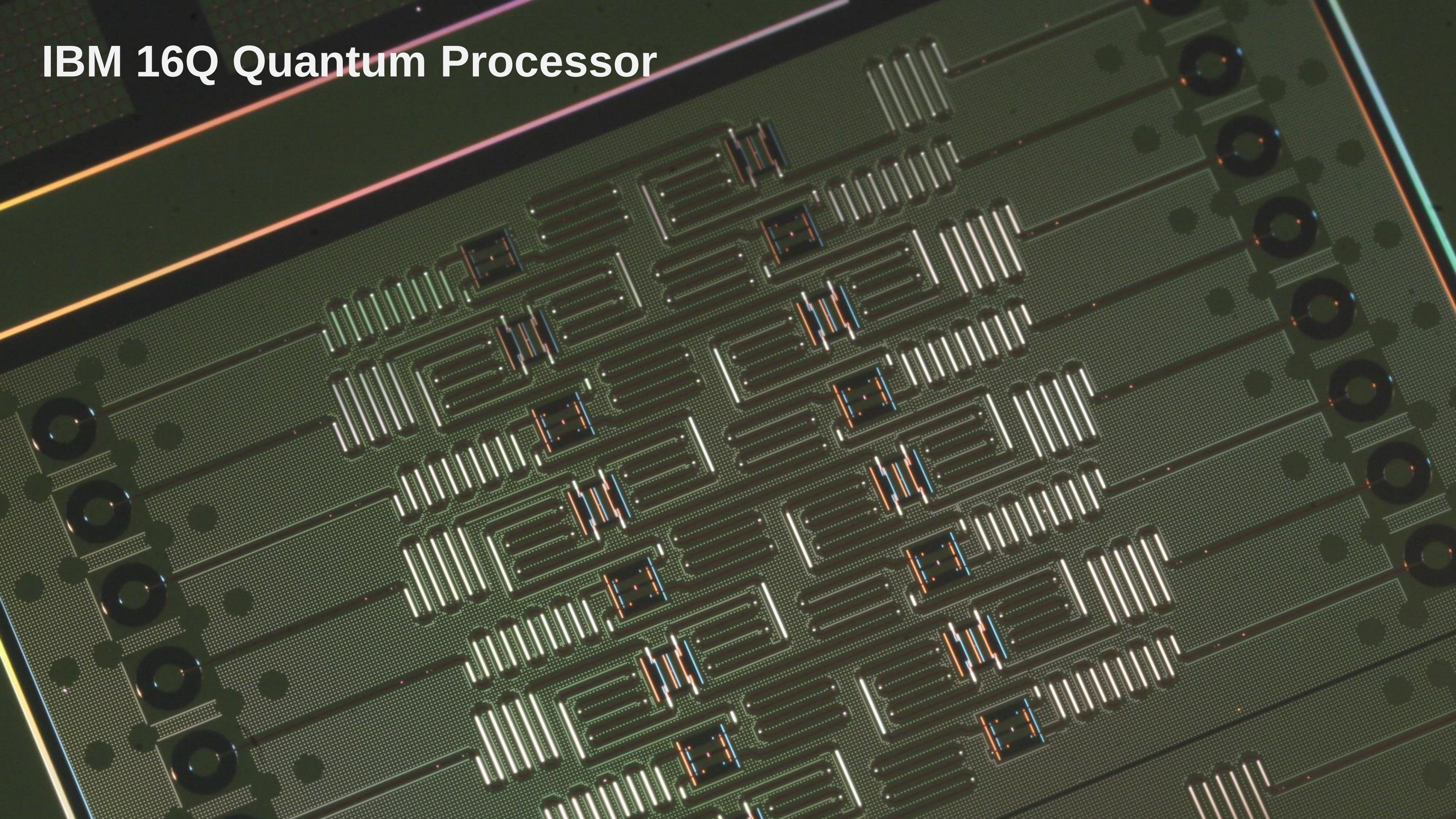
qubits



Average measurements

Frequency (GHz)	5.26
T1 (μs)	44.60
T2 (μs)	21.40
Gate error (10^{-3})	2.21
Readout error (10^{-2})	5.76

IBM 16Q Quantum Processor

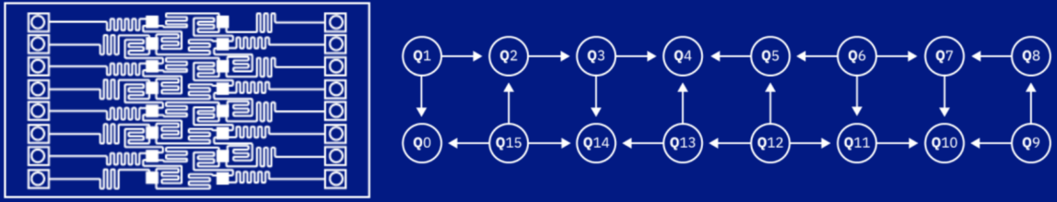


Operating metrics for the IBM Q Experience quantum computers are publicly available

Backend: ibmqx5 (16 Qubits)

ACTIVE

AVAILABLE ON QISKIT



Last Calibration: 2018-04-10 07:35:35

Fridge Temperature: 0.0138206 K

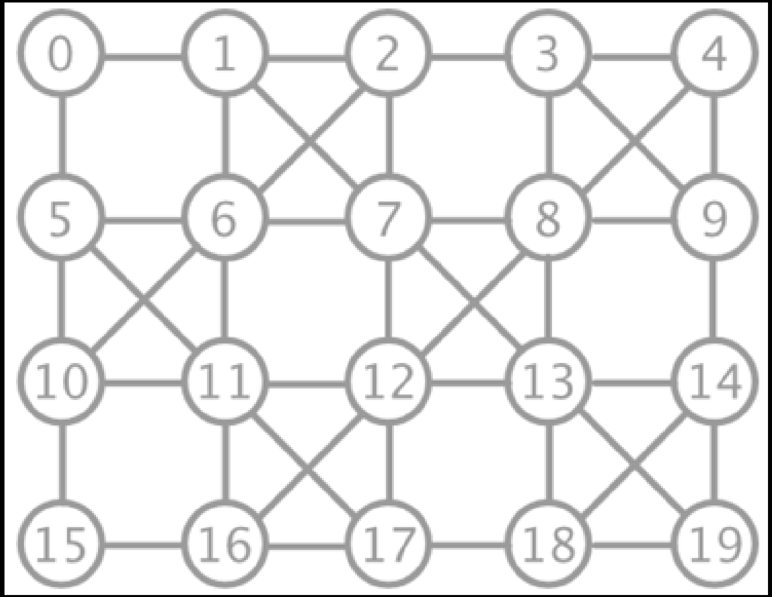
More details

	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Frequency (GHz)	5.26	5.40	5.28	5.08	4.98	5.15	5.31	5.25	5.12
T1 (μs)	23.70	39.10	40.40	52.50	49.20	42.60	51.90	43.30	41.50
T2 (μs)	27.60	61.80	64.30	92.30	91.50	45.10	68.40	58.60	76.50

IBM Q 20 Tokyo

20

qubits

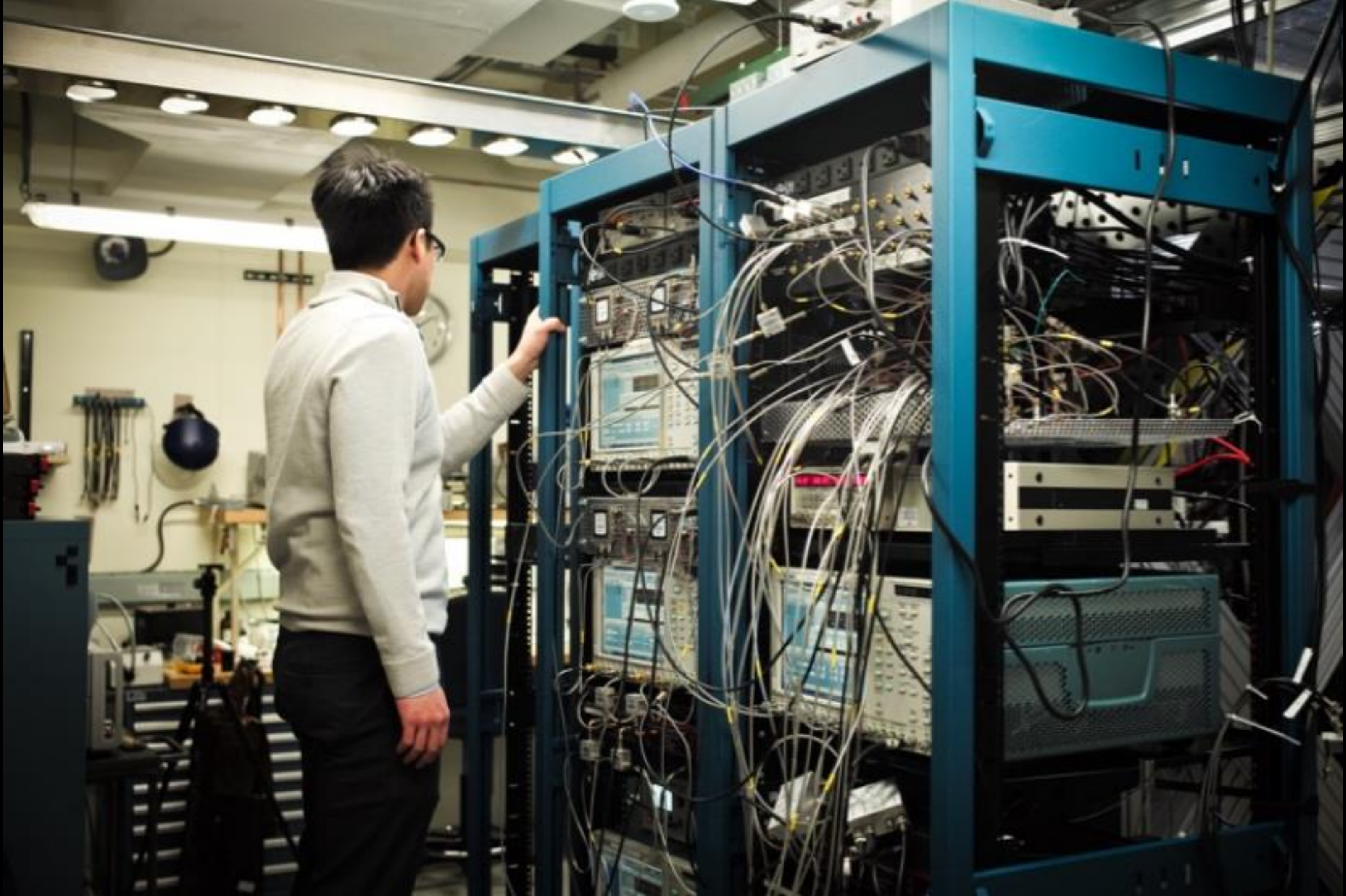


Average measurements

Frequency (GHz)	4.97
T1 (μs)	82.04
T2 (μs)	59.09
Gate error (10^{-3})	1.66
Readout error (10^{-2})	8.42

For IBM Q Network clients

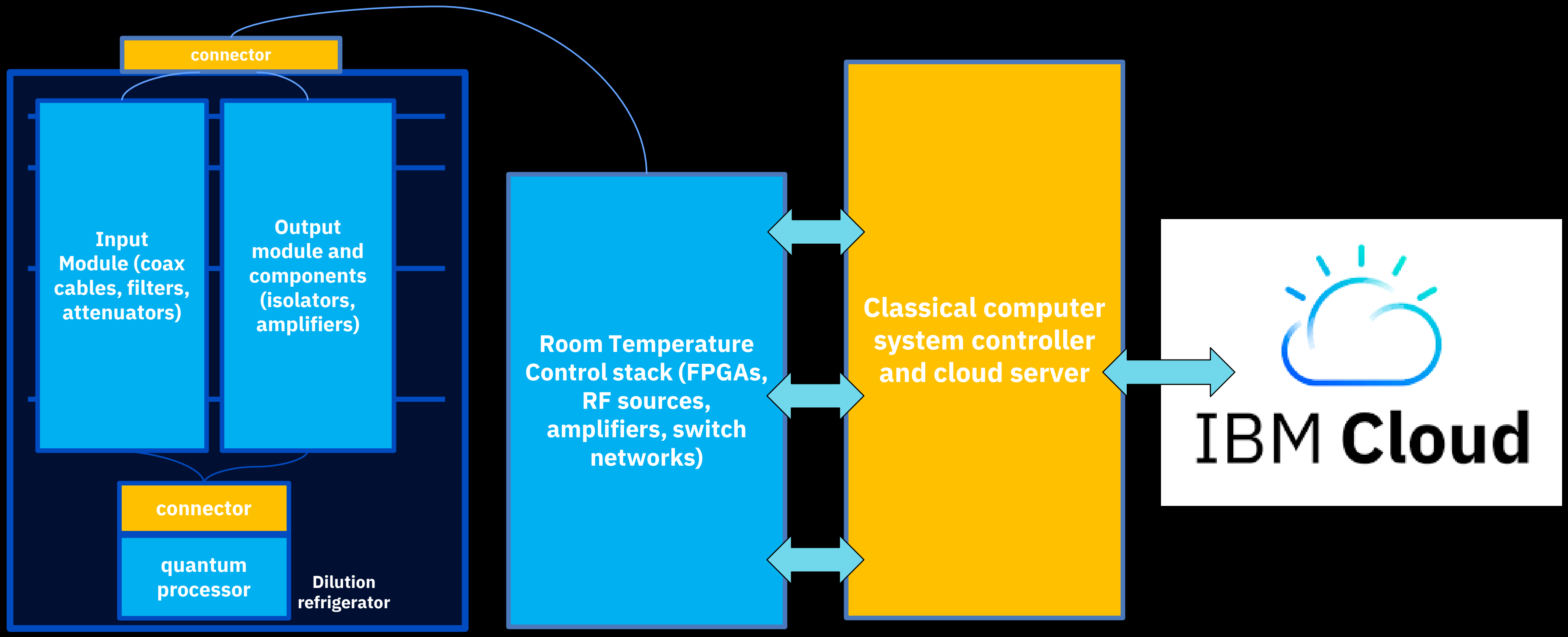

IBM Q




IBM Q


IBM Q
IBM Q
IBM Q

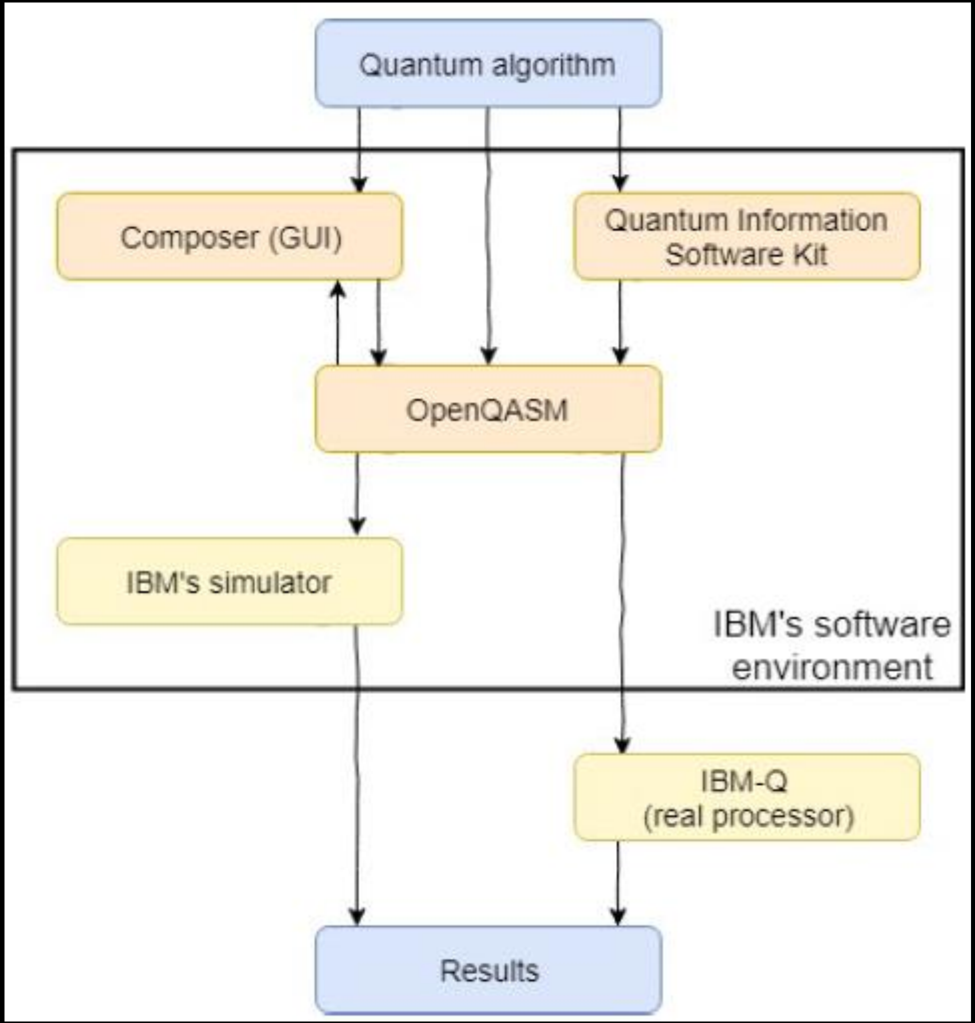
Schematic of the Complete IBM Q Hardware System



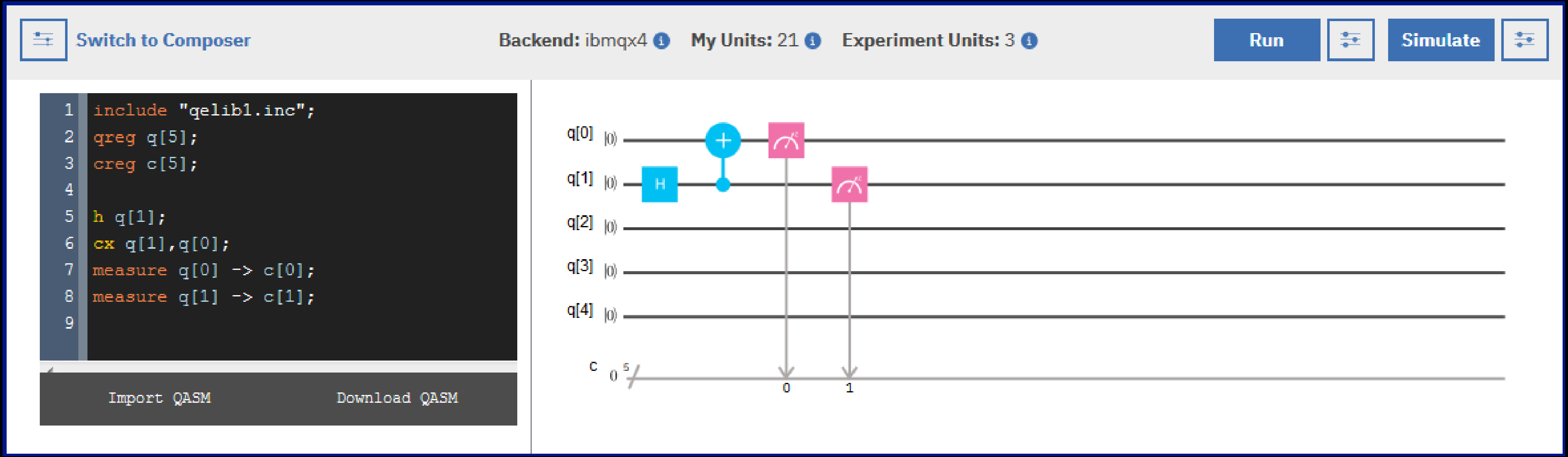
The hardware system

software…

IBM-Q Software Stack



IBM Q Experience – free, online



https://quantumexperience.ng.bluemix.net/qx/editor

Quantum programs for the 5 qubit machine can be constructed visually and then either simulated or run on the hardware.

Open Quantum Assembly Language

Andrew W. Cross, Lev S. Bishop, John A. Smolin, Jay M. Gambetta

(Submitted on 11 Jul 2017 (v1), last revised 13 Jul 2017 (this version, v2))

https://arxiv.org/abs/1707.03429
https://github.com/QISKit/openqasm

OpenQASM features

- Define quantum and classical **registers**: `qreg qr[8]; creg cr[8];`
- Apply **built-in unitary** operations **U** and **CX**: `U(pi/2,0,pi) qr[0]; CX qr[0],qr[1];`
- Define additional gates as **subroutines** using combinations of **U** and **CX**:

```
gate swap a,b {  //swap the quantum states of qubits a and b
      CX a,b;
      CX b,a;
      CX a,b;
}
```

- **Include** subroutines defined in other files: `include "qelib1.inc"`
- Perform **register-level** operations: `h qr; CX qra,qrb;`
- **Measure** qubits: `measure qr[0] -> cr[0];`
- Use **barriers** to limit compiler optimizations: `x qr[0]; barrier qr[0]; x qr[0];`
- Apply classically **conditioned** operations: `if (cr[0]==1) { x qr[1]; }`

```
include "qelib1.inc";
qreg q[2];
creg c[2];

h q[0];
cx q[0],q[1];
t q[1];
cx q[0],q[1];
measure q[0] -> c[0];
measure q[1] -> c[1];
```

- **Flexible, backend-agnostic, intermediate representation of instructions to be run on quantum hardware or simulator**
- **Not suitable for writing complex programs by hand → *use SDK to generate!***

QISKit SDK

- **QISKit** is an open source software development kit
- **QISKit** provides libraries, documentation, a simulator, and connections to IBM Q devices
- **QISKit** is easily extensible

https://qiskit.org/

QISKit on github

https://github.com/qiskit/

Repositories 19 | People 30 | Projects 0

Pinned repositories

qiskit-tutorial
A collection of Jupyter notebooks using Qiskit
Jupyter Notebook ★ 495 Forks 252

qiskit-terra
Terra provides the foundations for Qiskit. It allows the user to write quantum circuits easily, and takes care of the constraints of real hardware.
Python ★ 2.2k Forks 655

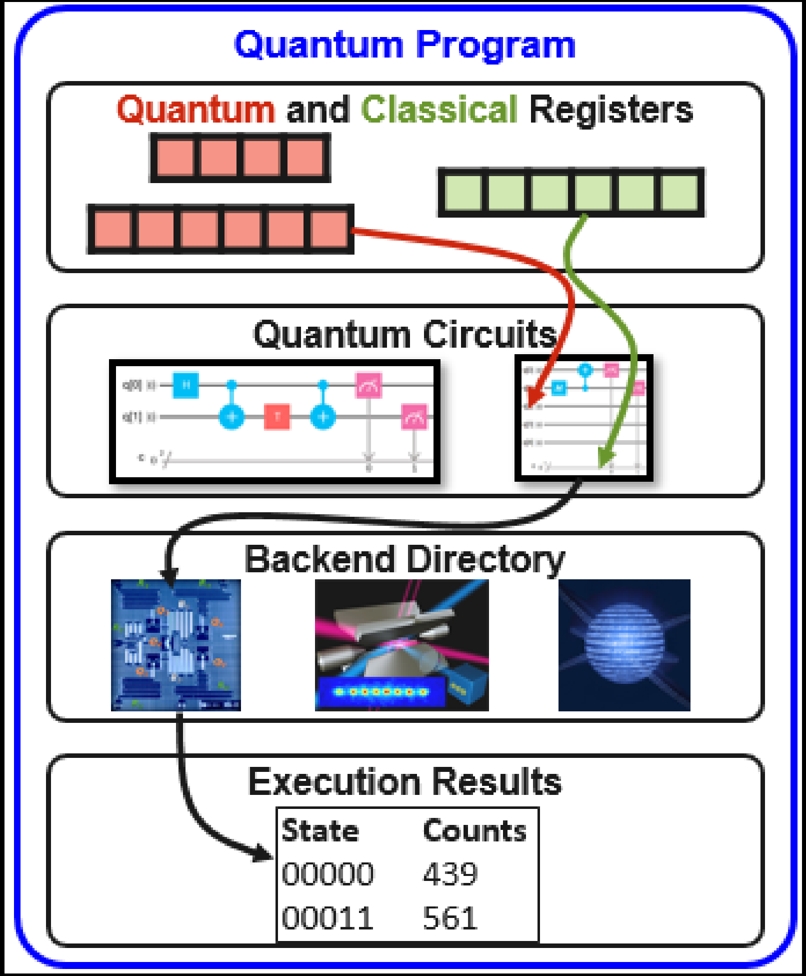
openqasm
Gate and operation specification for quantum circuits
TeX ★ 260 Forks 88

qiskit-presentations
Awesome Qiskit presentations
Jupyter Notebook ★ 13 Forks 6

aqua
Aqua provides a library and tools to build applications for Noisy Intermediate-Scale Quantum (NISQ) computers.
Python ★ 91 Forks 55

ibmq-device-information
Information about the different remote backends available for qiskit users with a IBMQ account
★ 108 Forks 56

QISKit SDK basic flow



QISKit SDK – defining circuit

- Let's create the 2-qubit Bell's state using QISKit
- The circuit can be visualized in a publication quality Latex-based diagrams

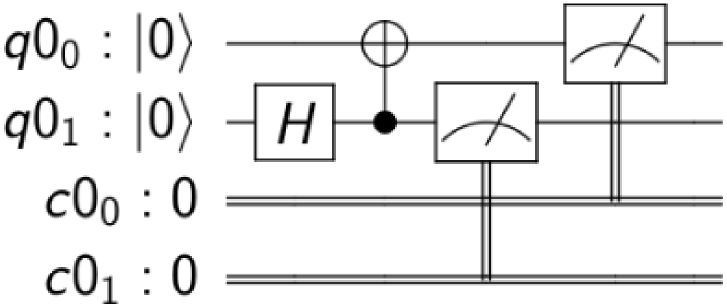
In [1]:
```
from qiskit import QuantumCircuit
from qiskit import ClassicalRegister, QuantumRegister
from qiskit import execute

#setup
qr = QuantumRegister(2)
cr = ClassicalRegister(2)
circuit = QuantumCircuit(qr, cr)
```

In [2]:
```
circuit.h(qr[1])
circuit.cx(qr[1], qr[0])
circuit.measure(qr, cr)
```

Out[2]:
```
<qiskit._instructionset.InstructionSet at 0x56b1390>
```

In [3]:
```
from qiskit.tools.visualization import circuit_drawer
circuit_drawer(circuit)
```

Out[3]:



QISKit SDK
define backend and run

```
from qiskit import IBMQ
```

```
#Earlier IBMQ.save_account('TOKEN') needs to be run
IBMQ.load_accounts()
```

```
backend=IBMQ.get_backend('ibmqx4')
```

```
job = execute(circuit, backend=backend, shots=1024,
                max_credits=3)
```

```
from qiskit import Aer
backend = Aer.get_backend('qasm_simulator')
job = execute(circuit, backend)
```

Simulator

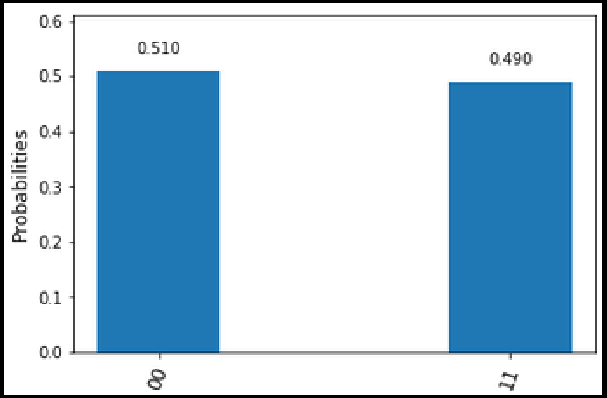
```
job.status()
```

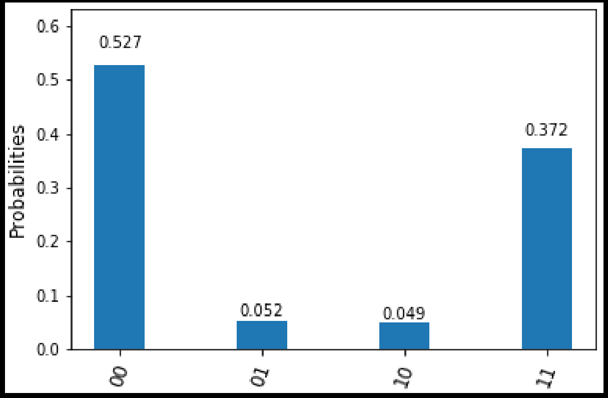
```
<JobStatus.DONE: 'job has successfully run'>
```

```
result = job.result()
```

```
from qiskit.tools.visualization import plot_histogram
plot_histogram(result.get_counts(circuit))
```

Real device





QISKit - monitoring job status

```
import time

lapse = 0
interval = 60
while job_exp.status().name != 'DONE':
    print('Status @ {} seconds'.format(interval * lapse))
    print(job_exp.status())
    print(job_exp.queue_position())
    time.sleep(interval)
    lapse += 1
print(job.status())
```

```
Status @ 0 seconds
JobStatus.QUEUED
39
Status @ 60 seconds
JobStatus.QUEUED
39
Status @ 120 seconds
JobStatus.QUEUED
39
```

QISKit - managing jobs

Listing jobs on given backend:

```
backend=IBMQ.get_backend('ibmqx4')
```

```
for ran_job in backend.jobs(limit=5):
    print(str(ran_job.job_id()) + " " + str(ran_job.status()))
```

```
5be9510117436b0052751cc8 JobStatus.QUEUED
5be70d5b054f3d005ae77a6c JobStatus.CANCELLED
5be706cfa9ff0f0053fa23a3 JobStatus.CANCELLED
5be705c6846b1b0052e14fc6 JobStatus.CANCELLED
```

Cancelling job:

```
job = backend.retrieve_job('5be9510117436b0052751cc8')
```

```
job.cancel()
```

QISKit evolution

- 4 directions (elements) of quantum software ecosystem
- **Terra** (earth) – fundation on which the rest of the software lies
- **Aqua** (water) – allowing to find real world applications using QC as accelerators for specific computational tasks

- **Ignis** (fire) – fighting noise and errors, improving gates, etc.
- **Aer** (air) – simulators, emulators, debuggers

QISkit Aer

- Aer provides set of simulators
- For example unitary simulator
- Currently IBM-Q hosted simulator supports up to 32 qubits

```
from qiskit import Aer

Aer.backends()
```

```
[<QasmSimulatorPy('qasm_simulator_py') from Aer()>,
 <StatevectorSimulatorPy('statevector_simulator_py') from Aer()>,
 <UnitarySimulator('unitary_simulator') from Aer()>]
```

```
from qiskit import Aer
backend = Aer.get_backend('unitary_simulator')
job = execute(circuit, backend)
```

```
job.status()
```

```
<JobStatus.DONE: 'job has successfully run'>
```

```
import numpy as np
np.round(job.result().get_data(circuit)['unitary'], 3)
```

```
array([[ 0.707+0.j,  0.   +0.j,  0.707-0.j,  0.   +0.j],
       [ 0.   +0.j,  0.707+0.j,  0.   +0.j,  0.707-0.j],
       [ 0.   +0.j,  0.707+0.j,  0.   +0.j, -0.707+0.j],
       [ 0.707+0.j,  0.   +0.j, -0.707+0.j,  0.   +0.j]])
```

```
IBMQ.backends()
```

```
[<IBMQBackend('ibmqx4') from IBMQ()>,
 <IBMQBackend('ibmqx5') from IBMQ()>,
 <IBMQBackend('ibmqx2') from IBMQ()>,
 <IBMQBackend('ibmq_16_melbourne') from IBMQ()>,
 <IBMQBackend('ibmq_qasm_simulator') from IBMQ()>]
```

```
backend = IBMQ.get_backend('ibmq_qasm_simulator')
```

QISkit Ignis

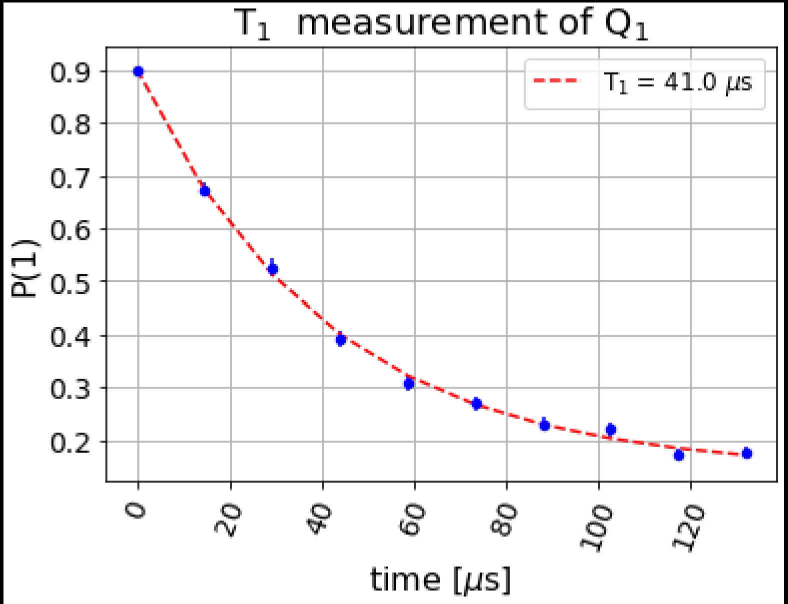
QISKit provides access to low level device characteristics

We can measure for example relaxation time T_1

Follow this tutorial for exact code: https://nbviewer.jupyter.org/github/Qiskit/qiskit-tutorial/blob/master/qiskit/ignis/relaxation_and_decoherence.ipynb

```python
from qiskit import QuantumCircuit
from qiskit import ClassicalRegister, QuantumRegister
from qiskit import execute
from qiskit import IBMQ

backend = IBMQ.get_backend('ibmq_16_melbourne') # the device to run on

params = backend.properties()['qubits'][1]
pulse_length=params['gateTime']['value'] # single-qubit gate time
buffer_length=params['buffer']['value'] # spacing between pulses
unit = params['gateTime']['unit']

print('Qubit 1, single gate length: '
      + str(round(pulse_length,2)) + ' ' + unit)
print('Qubit 1, buffer between gates: '
      + str(round(buffer_length,2)) + ' ' + unit)
```

```
Qubit 1, single gate length: 100 ns
Qubit 1, buffer between gates: 10 ns
```



QISKit AQUA

No need to know quantum cirquits. Domain experts can work in their existing frameworks without disruption

Extensible to multiple domains

AQUA Chemistry

Interfaces for:

- » **Gaussian 16**
- » **PSI4**
- » **PySCF**
- » **PyQuante**

AQUA Artificial Intelligence

AQUA Optimization

https://qiskit.org/aqua

Final Variational Parameters

```
[-1.70505133  5.08016856
  6.13194571  4.12524575
 -0.26812628  3.90229532
  5.60418357 -3.50842279
 -3.92937001  6.17154054
  5.55945014  5.89555566
 -4.83748112 -5.56210097
  5.58561689 -5.99641163
  1.65834199 -1.49309666
 -2.47010646  0.4242964
  0.39126571  0.46051075
  4.28488151 -5.30498883]
```

Final Energy

```
-1.875 Hartree
```

Chemistry

Problem Specifications
- PSI4
- PyQuante

Translators
Fermionic Hamiltonian generator
Qubits Hamiltonian generator

Solver API

Methods
Variational Quantum Eigensolver
Phase Estimation
Dynamics
Grover

OpenQASM

QISKit API
Optimizing Transpiler

OpenPulse API

Hardware **Simulator**

QISKit
How to get started?

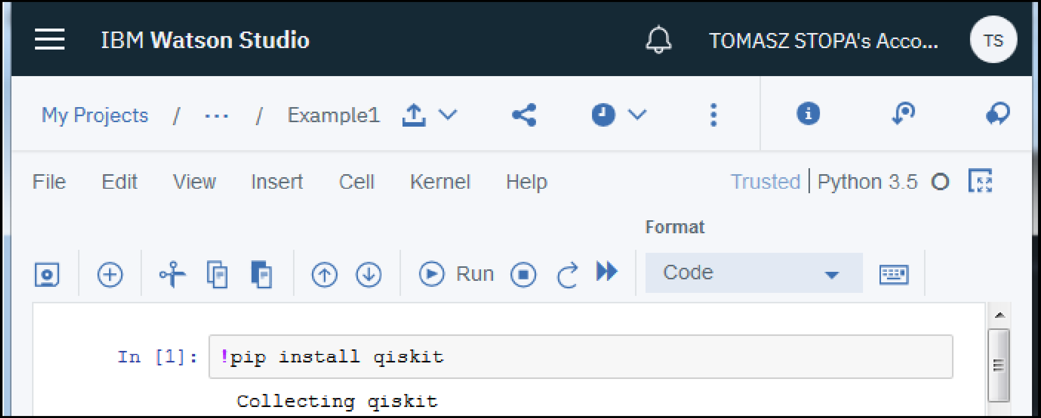
1. Watson Studio

IBM Cloud

Environment definitions ⊕ New environment definition

NAME	TOOL	HARDWARE CONFIGURATION	LANGUAGE	LAST MODIFIED	ACTIONS
Default Python 3.5 Free	Notebook	1 vCPU and 4 GB RAM	Python 3.5	16 Feb 2018	⋮

- Login to Watson Studio
- Create new Standard project
- Create runtime Environment
- Create Notebook
- Install QISKit (*!pip install qiskit*)
- Start writing in QISKit...



Takes ~5-10 minutes !

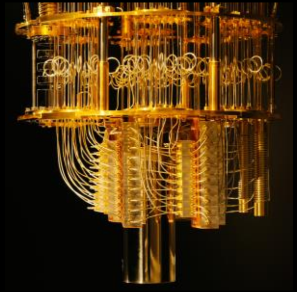
2. Your machine

- Install Python 3.5 or higher and Jupyter (for example using Anaconda distribution)
- Account on IBM Q Experience (for accessing IBM-Q devices)
- Run ***pip install qiskit*** to download and install the latest stable release and dependencies
- Start writing in QISKit…

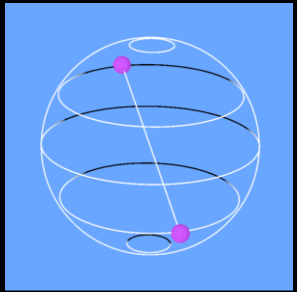
Takes ~1 hour

Your next steps to getting Quantum Ready

Discover more about IBM's quantum computing initiative



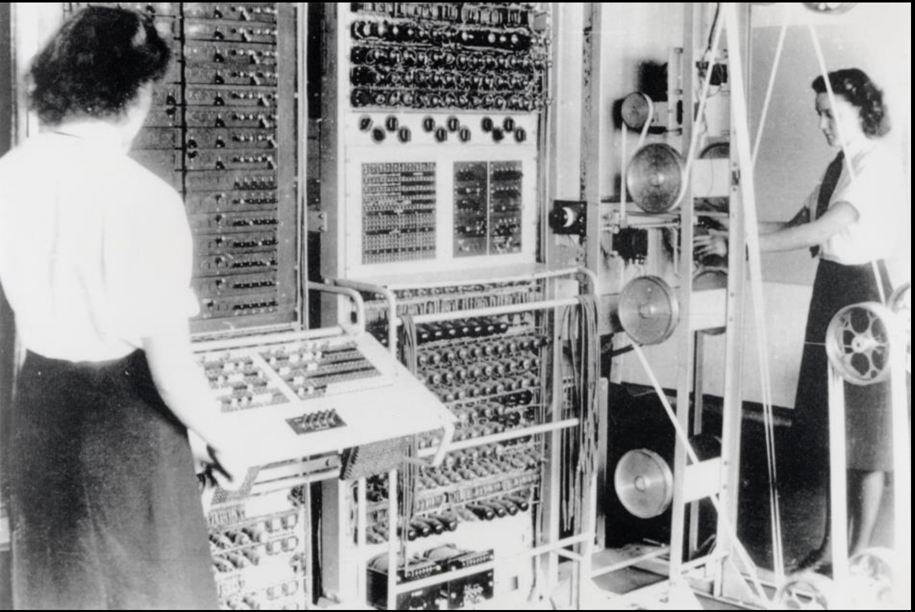
Explore the **IBM Q Experience** and start using real machines today

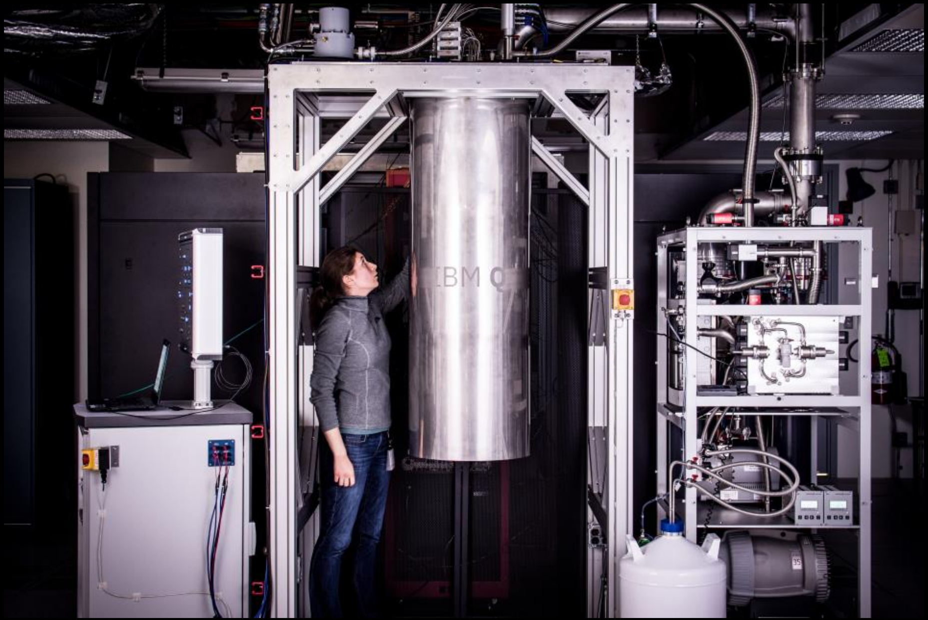


Learn about and start using the **QISKit** software development kit

Collaborate, research, and start applying quantum computing through the **IBM Q Network** future…

IBM Q





Colossus 1942
First electronic digital programmable computing device
With vacuum tubes, switches and plugs

IBM Q 2017
First universal quantum computing device
available to public

Where are we on the road to Quantum Advantage?

Quantum Foundations

Fundamentals of quantum information science

Create and scale qubits with increasing coherence

Create error detection and mitigation schemes

Quantum Ready

Core algorithm development

Increase quantum volume

Standardize performance benchmarks

System infrastructure and software enablement

Launch of IBM **Q** Experience

2016

2020s

Quantum Advantage

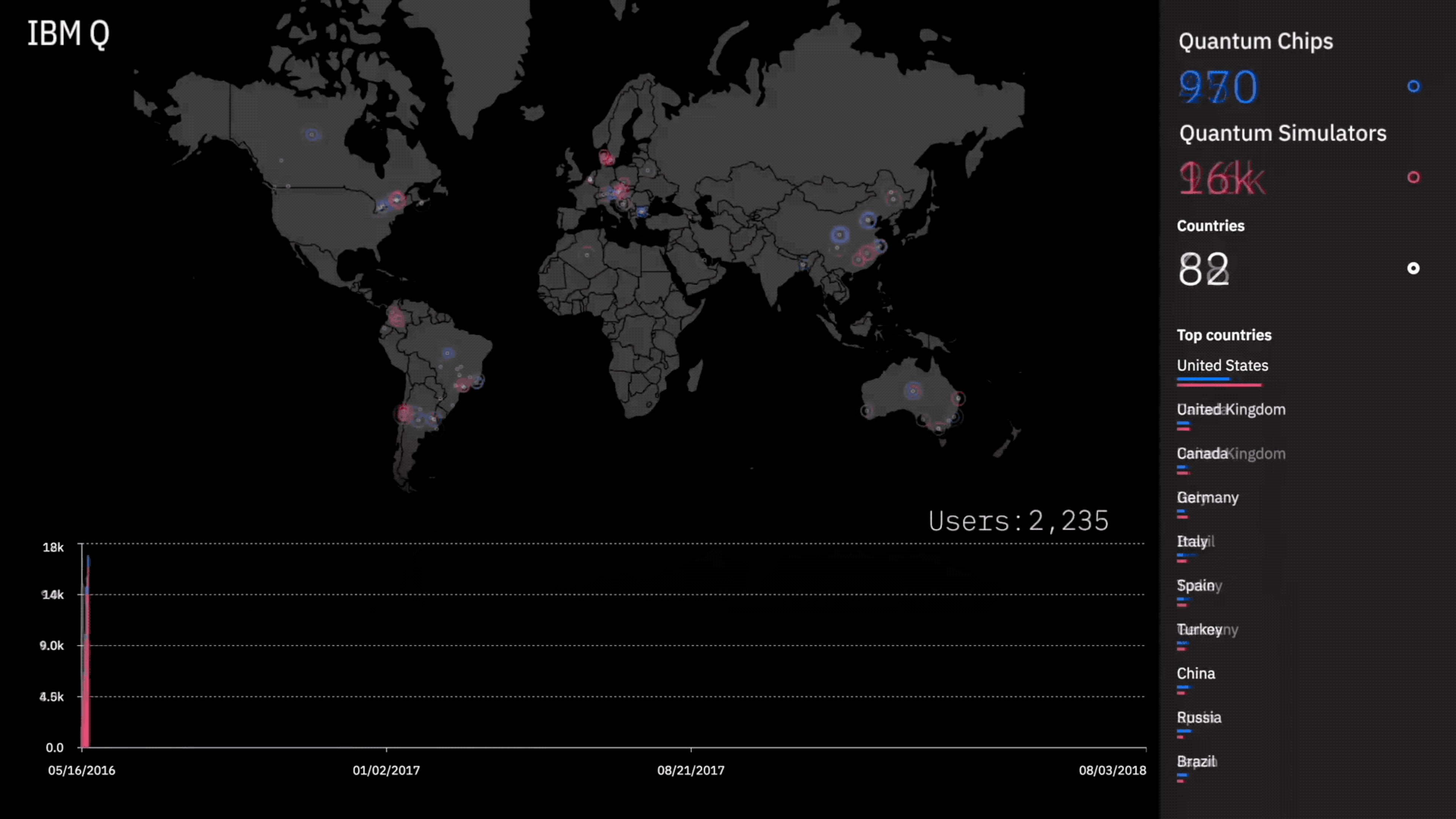
Demonstrate an advantage to using QC for real problems of interest

Extract Commercial Value

Enable scientific discovery

~1900

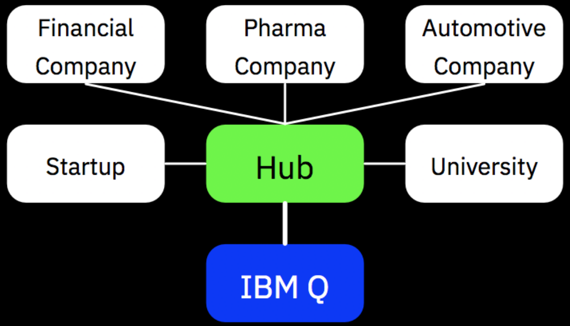
Today


IBM Q
Quantum Chips
970
Quantum Simulators
16k
Countries
82
Top countries
United States
United Kingdom
Canada
Germany
Italy
Spain
Turkey
China
Russia
Brazil
Users:2,235
18k
14k
9.0k
4.5k
0.0
05/16/2016
01/02/2017
08/21/2017
08/03/2018

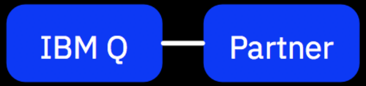
Next steps...

- Prototype of 50 qubit system being developed
- In future to be available for IBM Q Network users
- Simulator hosted on IBM Q optimized for POWER processors with GPUs → preliminary results show **10x performance gain** (https://www.ibm.com/blogs/research/2018/05/quantum-circuits/)

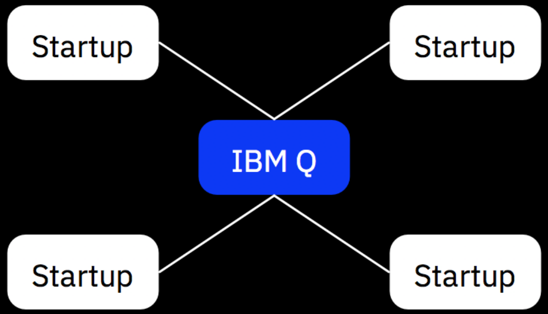
Further engagements within the IBM Q Network



Hubs
Regional centers of quantum computing R&D and ecosystem



Partners
Pioneers of quantum computing in a specific industry or academic field



Startups
Rapidly advance early applications

Possible early application areas for quantum computing

Chemistry

- Simulating reactions (drug discovery), molecules, proteins

Solid State Physics

- New materials
- Electronic structure calculation

Artificial Intelligence

- Classification, machine learning, linear algebra, image recognition

Financial Services

- Portfolio optimization, scenario analysis, pricing, risk nalysis (https://arxiv.org/abs/1806.06893)

Quantum Cryptography

Q Thanks!

Please note

IBM’s statements regarding its plans, directions, and intent are subject to change or withdrawal without notice and at IBM’s sole discretion.

Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision.

The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code or functionality. Information about potential future products may not be incorporated into any contract.

The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user’s job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

Notices and disclaimers

Information in these presentations (including information relating to products that have not yet been announced by IBM) has been reviewed for accuracy as of the date of initial publication and could include unintentional technical or typographical errors. IBM shall have no responsibility to update this information. **This document is distributed "as is" without any warranty, either express or implied. In no event, shall IBM be liable for any damage arising from the use of this information, including but not limited to, loss of data, business interruption, loss of profit or loss of opportunity.** IBM products and services are warranted per the terms and conditions of the agreements under which they are provided.

IBM products are manufactured from new parts or new and used parts. In some cases, a product may not be new and may have been previously installed. Regardless, our warranty terms apply.

Any statements regarding IBM's future direction, intent or product plans are subject to change or withdrawal without notice.

Performance data contained herein was generally obtained in a controlled, isolated environments. Customer examples are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual performance, cost, savings or other results in other operating environments may vary.

References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business.

Workshops, sessions and associated materials may have been prepared by independent session speakers, and do not necessarily reflect the views of IBM. All materials and discussions are provided for informational purposes only, and are neither intended to, nor shall constitute legal or other guidance or advice to any individual participant or their specific situation.

It is the customer's responsibility to insure its own compliance with legal requirements and to obtain advice of competent legal counsel as to the identification and interpretation of any relevant laws and regulatory requirements that may affect the customer's business and any actions the customer may need to take to comply with such laws. IBM does not provide legal advice or represent or warrant that its services or products will ensure that the customer follows any law.

Notices and disclaimers continued

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products about this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products. IBM does not warrant the quality of any third-party products, or the ability of any such third-party products to interoperate with IBM's products. **IBM expressly disclaims all warranties, expressed or implied, including but not limited to, the implied warranties of merchantability and fitness for a purpose.**

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents, copyrights, trademarks or other intellectual property right.

.