Google AI Quantum

Machine Learning no Mundo Quântico

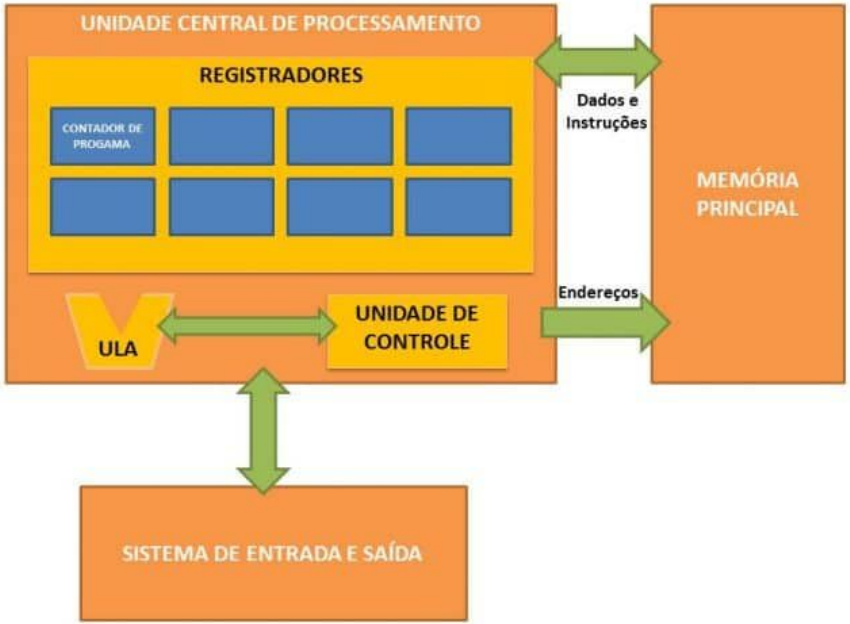
SPOILER: isso existe e não existe ao mesmo tempo!

Luciano Martins
Machine Learning Specialist

Agenda

- Review da “computação clássica”
- O “mundo quântico”
- Conceitos básicos da computação quântica
- O aprendizado de máquina quântico
- Cenário atual e próximos passos
- Referências

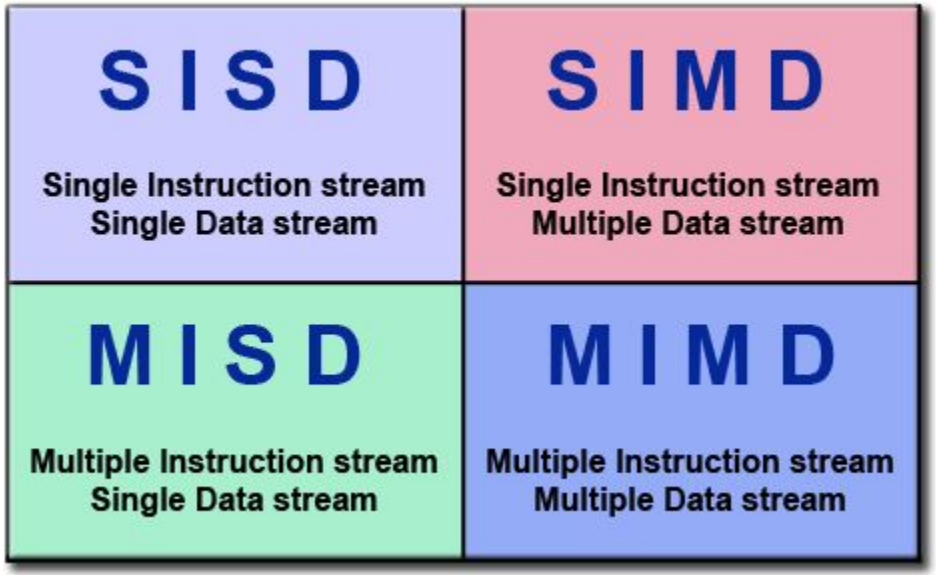
Introdução - Computação Clássica



Google Cloud

Arquitetura de Von Neumann (~1945)

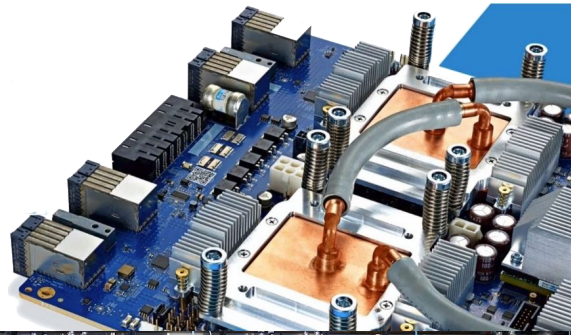
Introdução - Computação Paralela e Distribuída

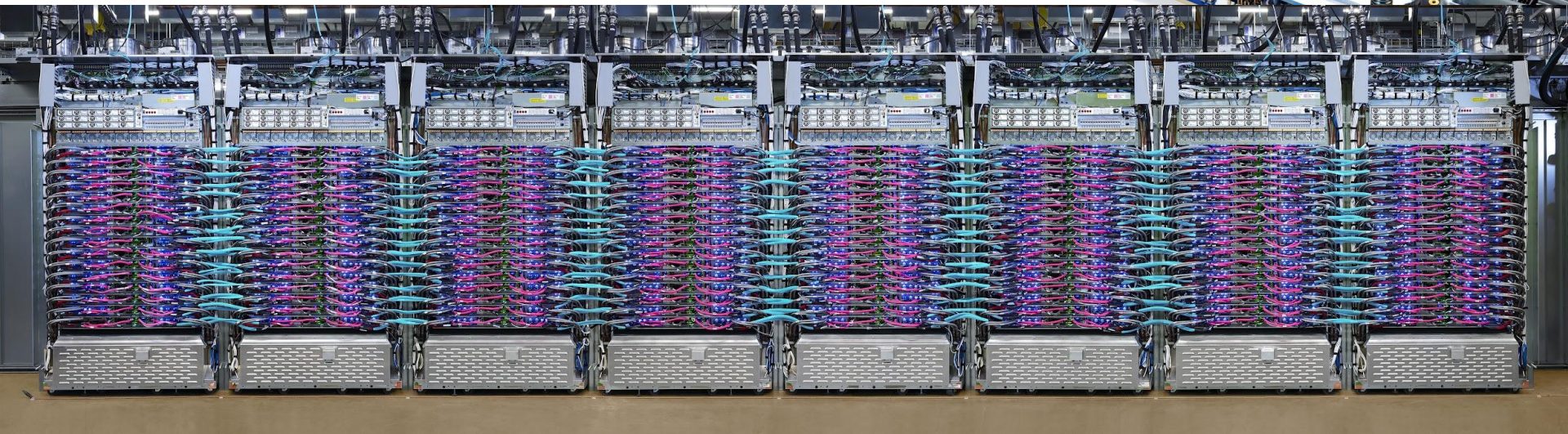


Taxonomia Clássica de Flynn (~ 1966)

Introdução - Computação Paralela e Distribuída

Google Cloud TPU v3
100 Petaflops de poder de aprendizado de máquina
= 100 bilhões de milhões
operações de ponto flutuante por segundo



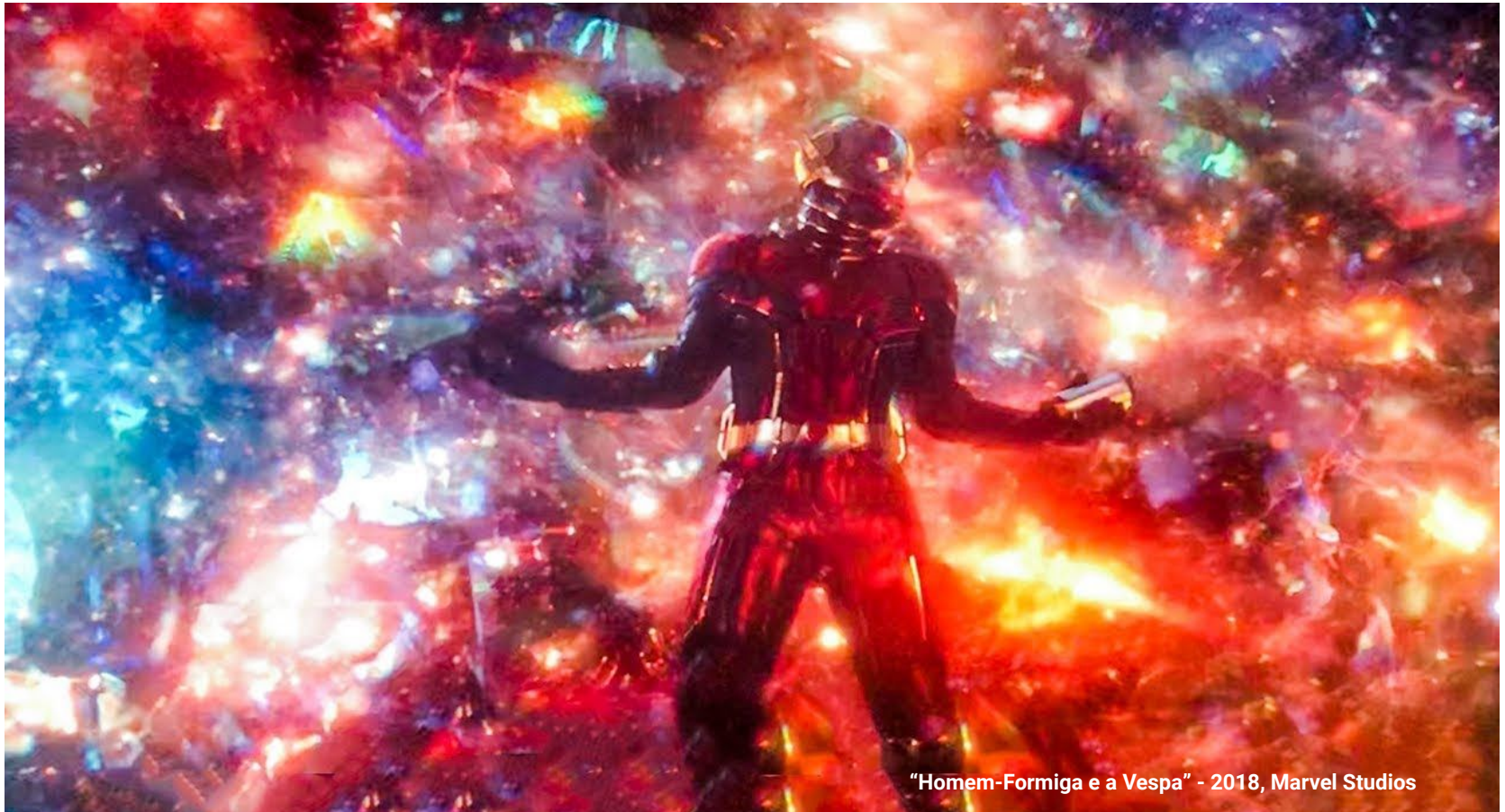


Introdução - Computação Clássica

Computadores clássicos, quando recebem tarefas, executam uma por vez

Google Cloud

O mundo quântico



“Homem-Formiga e a Vespa” - 2018, Marvel Studios

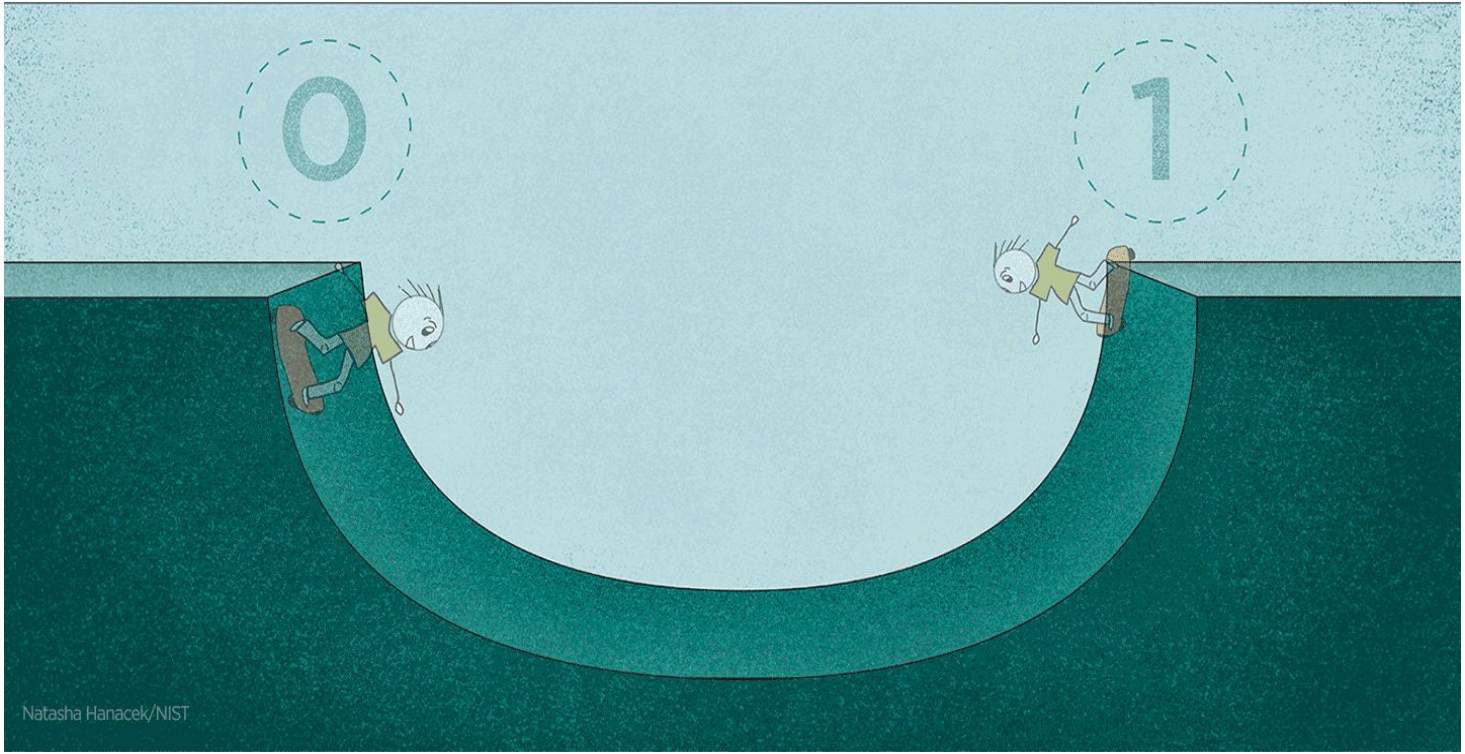
O mundo quântico

“Computadores quânticos são dispositivos que utilizam os fenômenos da mecânica quântica para executar algoritmos que não podem ser executados de forma eficiente em computadores clássicos (tradicionais).”

Left to right: *Max Planck, Albert Einstein, Niels Bohr, Louis de Broglie, Max Born, Paul Dirac*
Werner Heisenberg, Wolfgang Pauli, Erwin Schrödinger, Richard Feynman.

Google Cloud

O mundo quântico - Sobreposição (*Superposition*)



Google Cloud

Fonte: NIST

O mundo quântico - Sobreposição (*Superposition*)

300 bits vs. 300 qubits

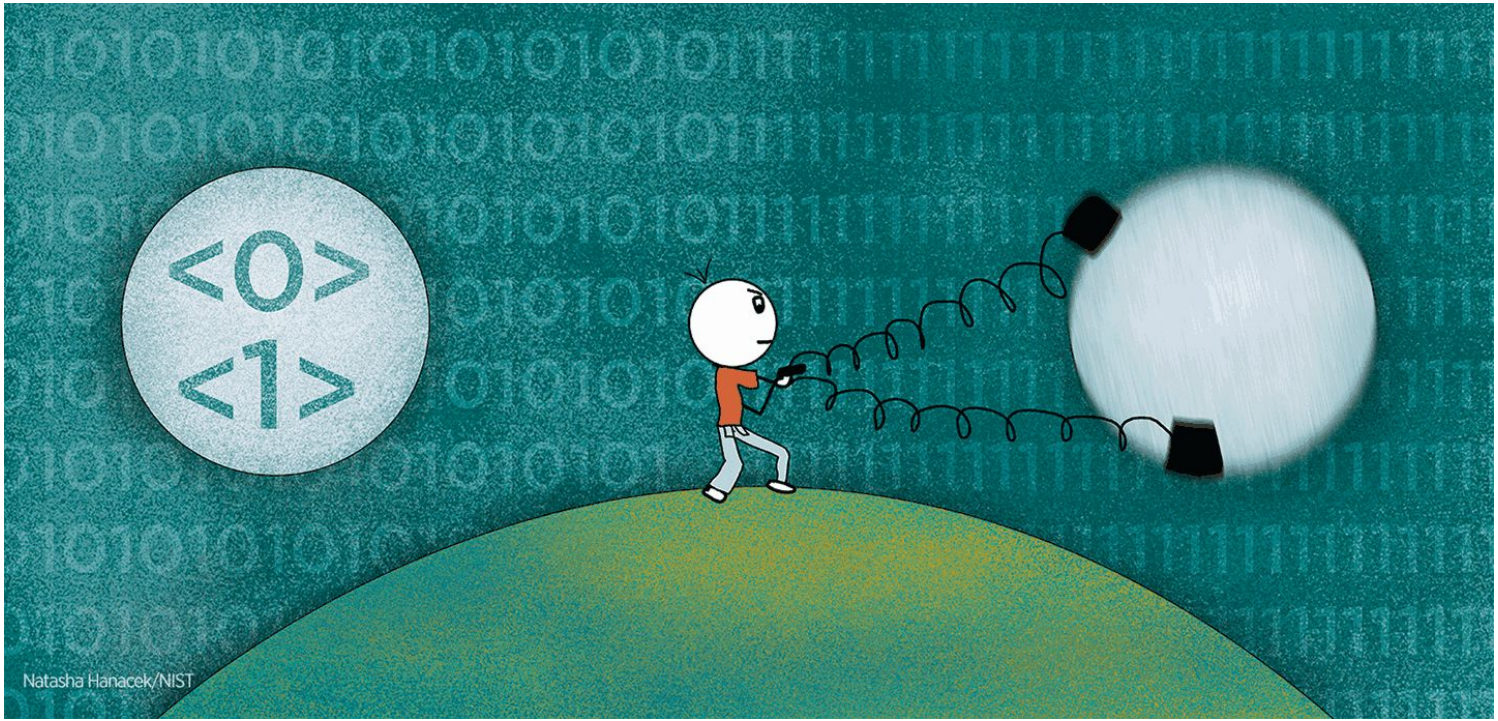
010001010110101001001110100010100101110100101001001010100100000001010111101
001010100101010101110100101010010010100101011101001010101010010010101010010101
010111010010010010100101010100100101001010010010100000100101010010100010101
101101010010001001000100001000101011110101001001010010110100101001001010101
11

2^300 ~ 10^90 estados (ou “bits” de informação)

(estima-se que haja ~10^80 partículas no universo)

Google Cloud

O mundo quântico - [De]Coerência ([De]*Coherence*)



Google Cloud

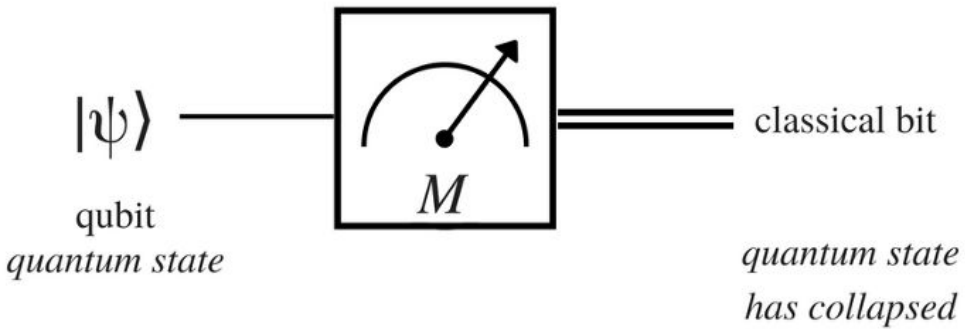
Fonte: NIST

O mundo quântico - Entrelaçamento (*Entanglement*)

Google Cloud

Fonte: NIST

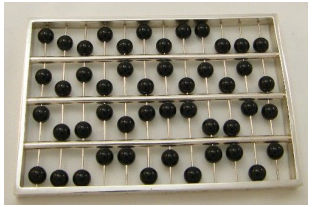
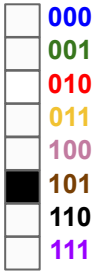
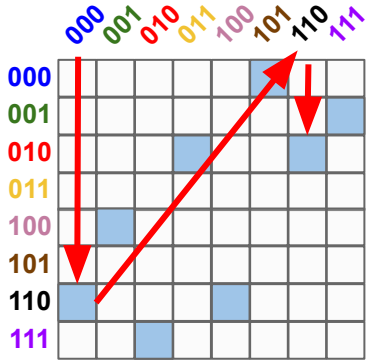
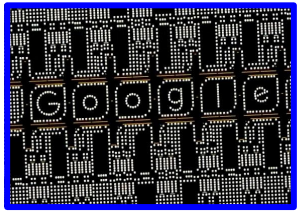
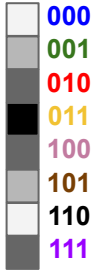
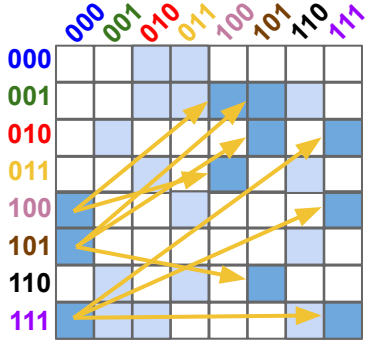
O mundo quântico - Medição quântica (*Measurement*)



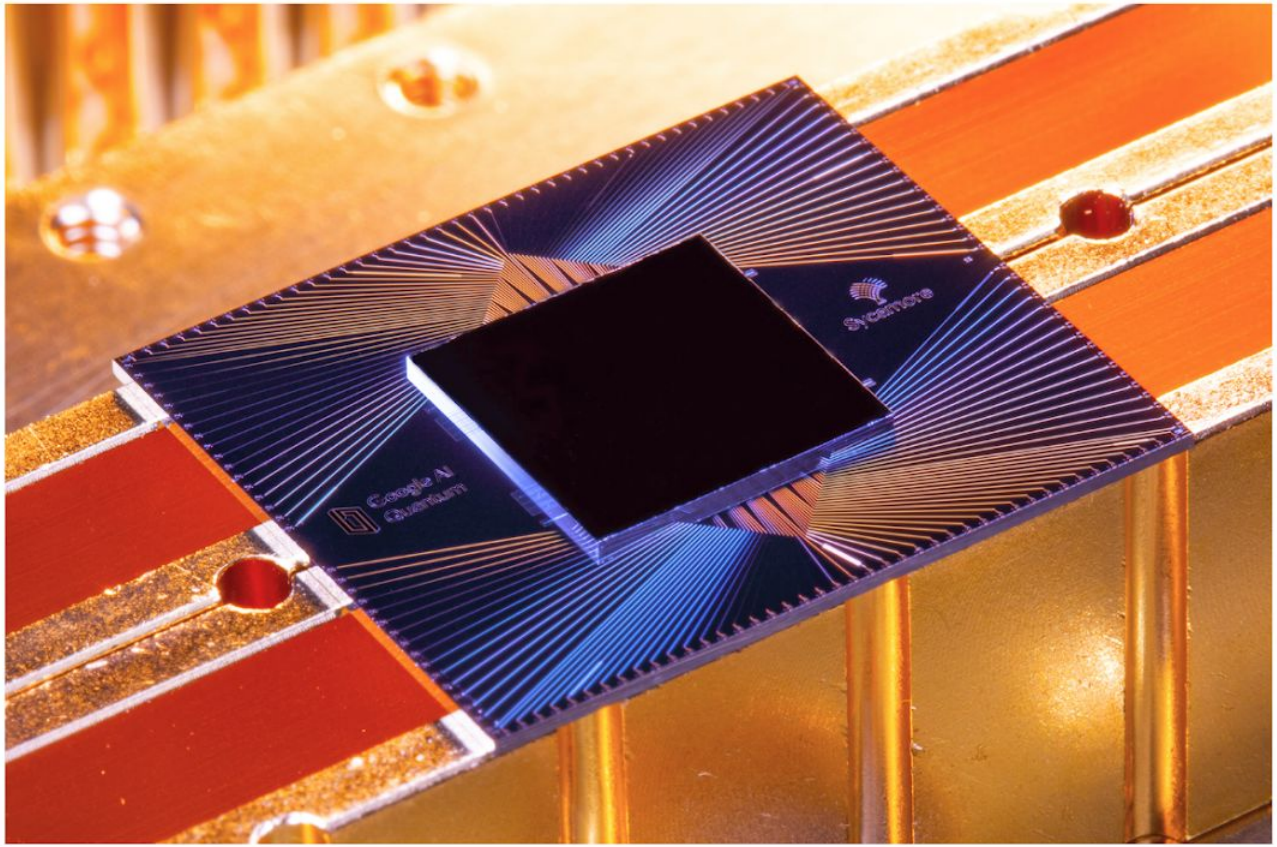
Em linhas gerais:

Uma operação em um qubit (algum tipo de estado de superposição de vetores de base | 0 não definido e | 1 não definido) para obter um bit clássico (o processo do qual é completamente aleatório)

A informação é composta pelos conceitos básicos

Física	1 bit	3 bits	Operações lógicas
 Clássica	0  1 	101  *Um ocupado de cada vez*	 Uma entrada por coluna
 Quântica	$c_0 0 + c_1 1$  0 1	$c_{000}000$ + $c_{001}001$ + $c_{010}010$ + $c_{011}011$ + ... + $c_{111}111$  Todos os 2^N estados clássicos têm amplitude simultaneamente	 Muitas entradas por coluna

Projetos Google Quantum (Sycamore)



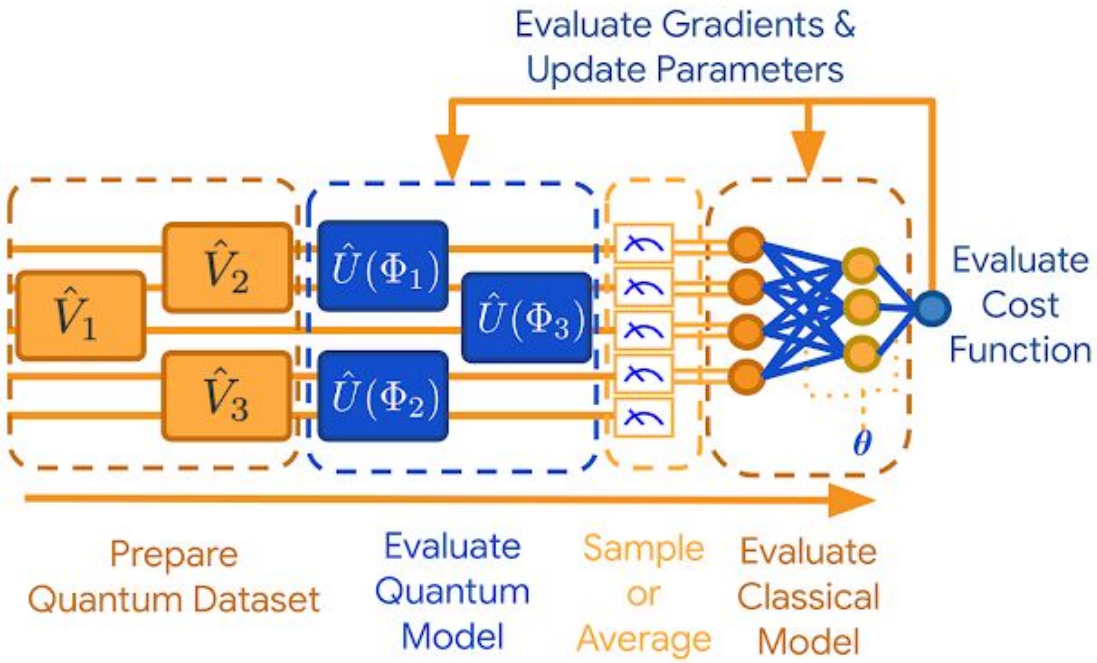
Os computadores quânticos

- São baseados em qubits em vez de bits
- Sua operação tem propriedades probabilísticas
- Pode existir em qualquer superposição de 2 ^ n estados
- Pode realizar operações paralelas em estados combinados
- Requer tratamento especial para obter resultados de computação
- Pode atingir uma aceleração exponencial em certos algoritmos clássicos

E o que seria Quantum Machine Learning (QML)?

- **Quantum Data** + **Modelos Híbridos** (Quântico-Clássicos)
- **Quantum Data:** Dados gerados por um “sistema quântico” natural ou artificial
- Exemplos de Quantum Data podem ser:
- Simulação química
- controle quântico
- metereologia quântica, etc
- **Modelos híbridos:** Possuem etapas processadas em CPU e em QPU
- Aplicado via a utilização de redes neurais quânticas (QNN)
- Busca otimizar a pesquisa de Machine Learning com redução de tempo e recursos
- Abordagem desta sessão: **Google Cirq** e **Tensorflow Quantum**

E o que seria Quantum Machine Learning (QML)?

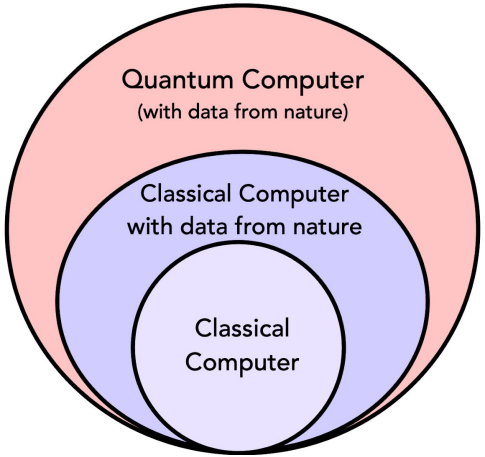


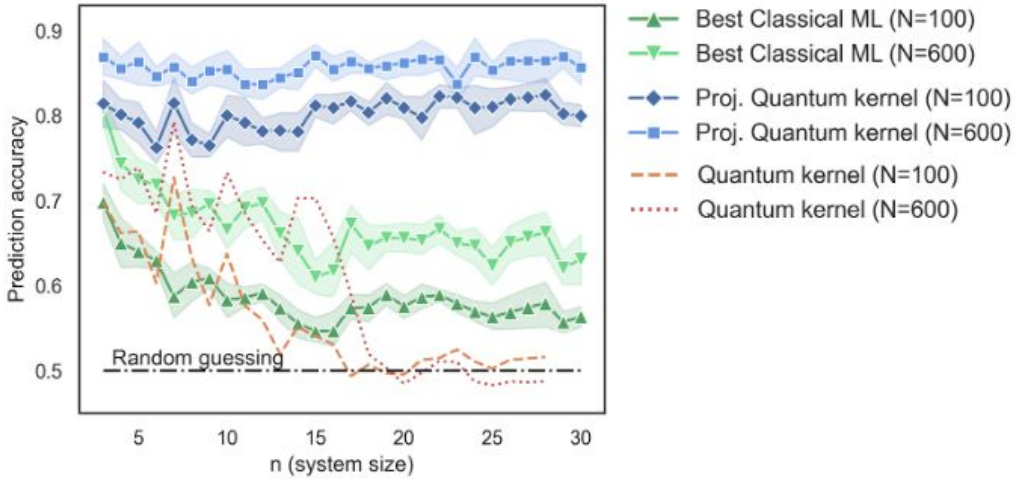
Google Cloud

E o que seria Quantum Machine Learning (QML)?

“Power of data in quantum machine learning” arXiv:2011.01938 [Feb, 2021]

Problems that could be solved by





Google Cloud

QML - Google Cirq

- Um framework opensource que auxilia na escrita, manipulação e otimização de circuitos quânticos
- Estes circuitos podem ser utilizados em computadores quânticos reais ou em simuladores
- Se baseia nos conceitos de **operators**, **gates**, **circuits** e **Simulations**

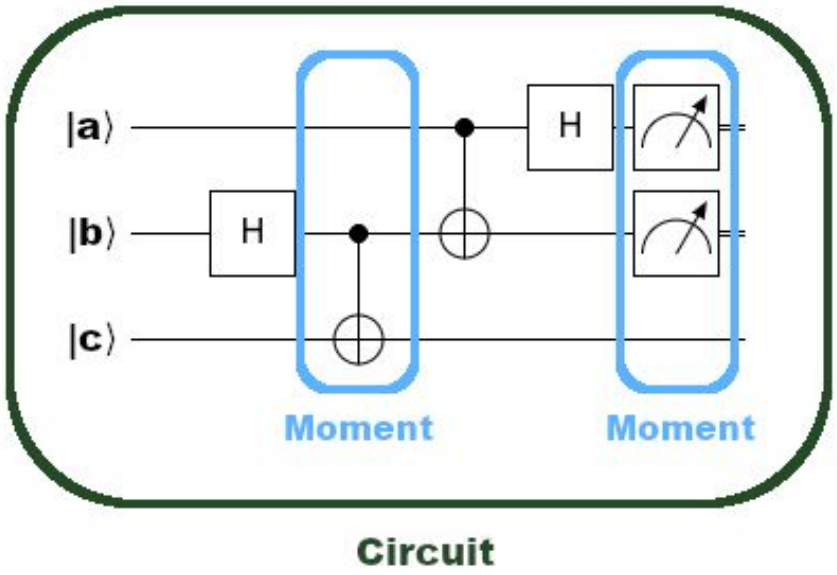
Google Cloud

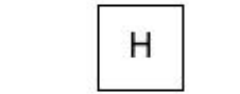
QML - Google Cirq

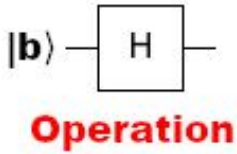
- Um **Gate** é um “efeito” aplicado à uma coleção de qubits
- O objeto resultado desta aplicação é uma **Operation**
- O **circuit** é uma coleção de **Moments** - um conjunto de **Operations** que acontecem num mesmo intervalo de tempo

QML - Google Cirq

Registradores
(Qubits)







Google Cloud

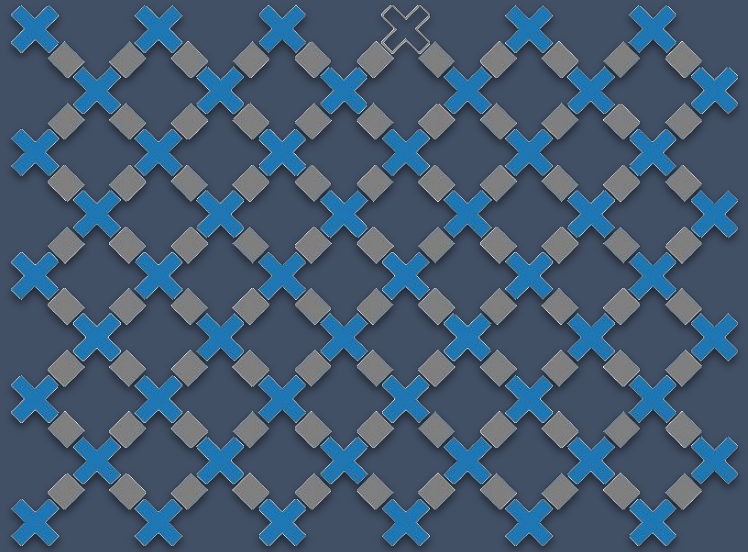
QML - Google Cirq

- Um framework opensource que auxilia na escrita, manipulação e otimização de circuitos quânticos
- Estes circuitos podem ser utilizados em computadores quânticos reais ou em simuladores
- Se baseia nos conceitos de **operators**, **gates**, **circuits** e **Simulations**

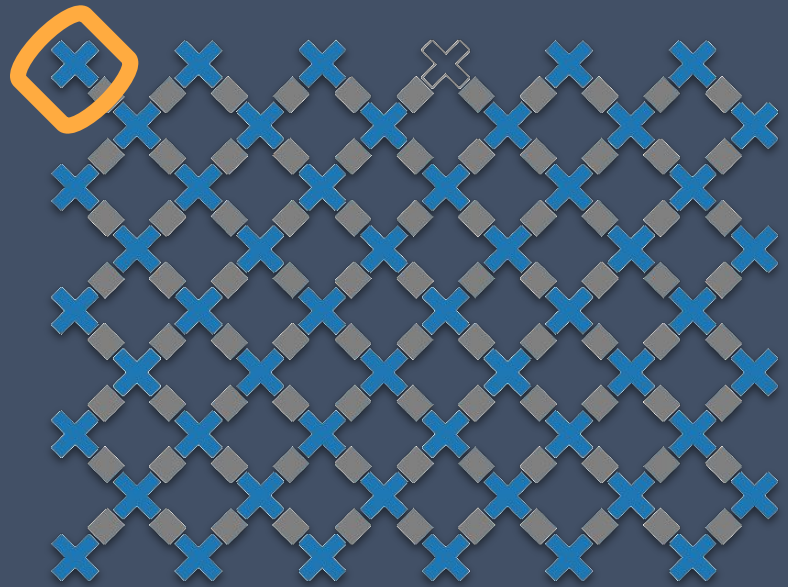
Google Cloud

```
import cirq

# Pick a qubit
qubit = cirq.GridQubit(0, 0)

# Create a circuit
circuit = cirq.Circuit(
    cirq.X(qubit)**0.5,  # Square root of NOT.
    cirq.measure(qubit, key='m')  # Measurement.
)
print("Circuit:")
print(circuit)

# Simulate the circuit several times.
simulator = cirq.Simulator()
result = simulator.run(circuit, repetitions=20)
print("Results:")
print(result)
```

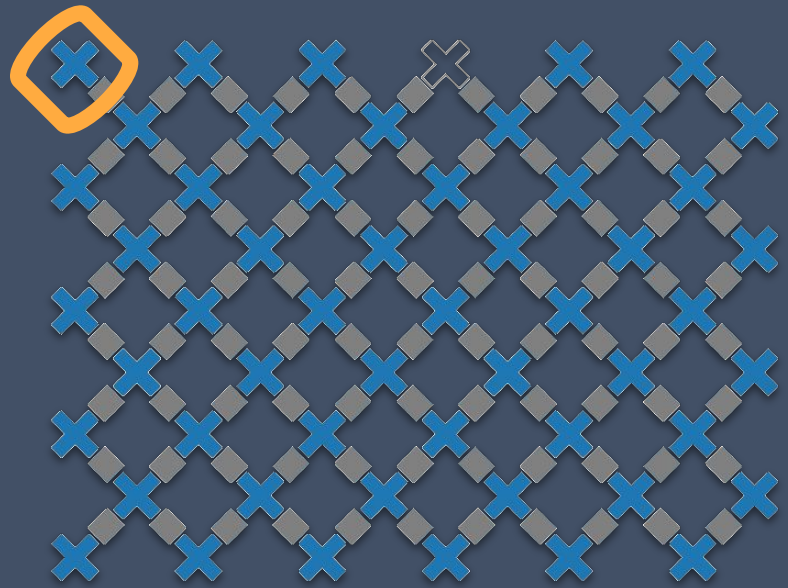
```
import cirq
import sympy
```



```
import cirq
import sympy

q0 = cirq.GridQubit(0, 0)
```

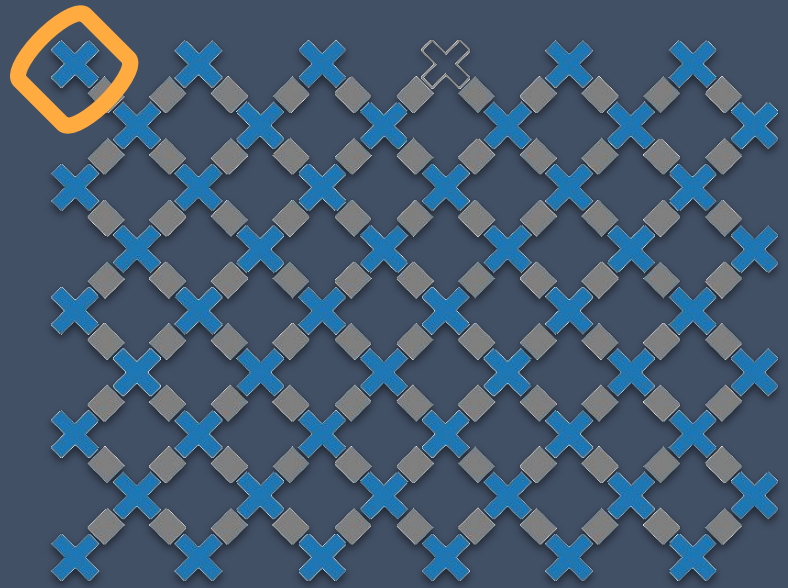


```
import cirq
import sympy

q0 = cirq.GridQubit(0, 0)

theta = sympy.Symbol('theta')
my_circuit = cirq.Circuit(cirq.Ry(theta)(q0))
```



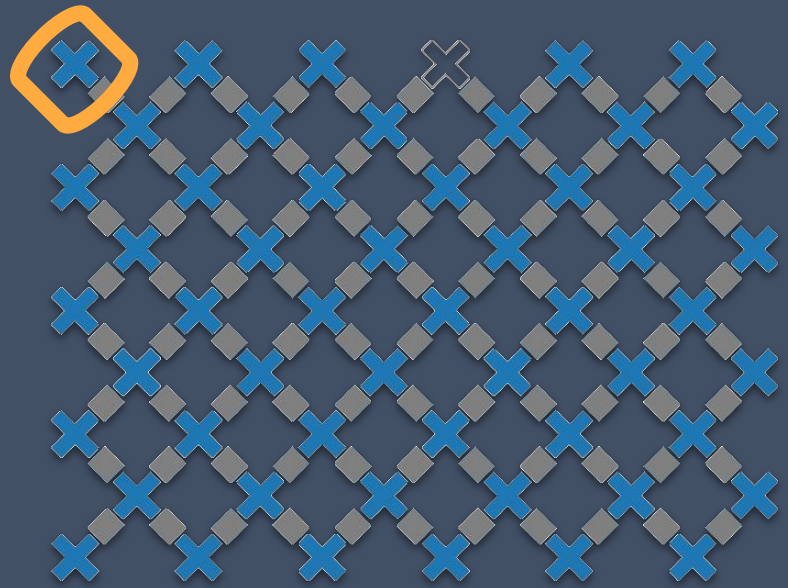
my_circuit

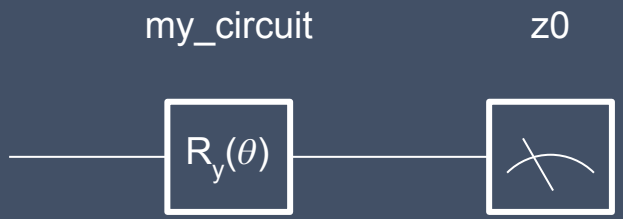
```
import cirq
import sympy

q0 = cirq.GridQubit(0, 0)

theta = sympy.Symbol('theta')
my_circuit = cirq.Circuit(cirq.Ry(theta)(q0))

param_resolver={theta: 1.2}
state_vector = cirq.Simulator().simulate(
     my_circuit, param_resolver).final_state
```



my_circuit

```
import cirq
import sympy

q0 = cirq.GridQubit(0, 0)

theta = sympy.Symbol('theta')
my_circuit = cirq.Circuit(cirq.Ry(theta)(q0))

param_resolver={theta: 1.2}
state_vector = cirq.Simulator().simulate(
     my_circuit, param_resolver).final_state

z0 = cirq.Z(q0)
z0.expectation_from_wavefunction(
     state_vector, qubit_map={q0: 0}).real
```





QML - Tensorflow Quantum

- Framework **híbrido** (quântico-clássico) baseado em **Tensorflow** e **Cirq**
- Auxilia na prototipação, treinamento, inferência e testes de modelos quânticos utilizando **quantum data**
- Fornece ferramentas para intercalar **algorítimos quânticos** e **lógica clássica**
- O treinamento é feito via a API **Keras**

Google Cloud

```
# A hybrid quantum-classical model.
model = tf.keras.Sequential([

    # Quantum circuit data comes in inside of tensors.
    tf.keras.Input(shape=(), dtype=tf.dtypes.string),

    # Parametrized Quantum Circuit (PQC) provides
    # output data from the input circuits run on a
    # quantum computer.
    tfq.layers.PQC(my_circuit,[cirq.Z(q1),cirq.X(q0)]),

    # Output data from qc passed through model.
    tf.keras.layers.Dense(50)

])
```

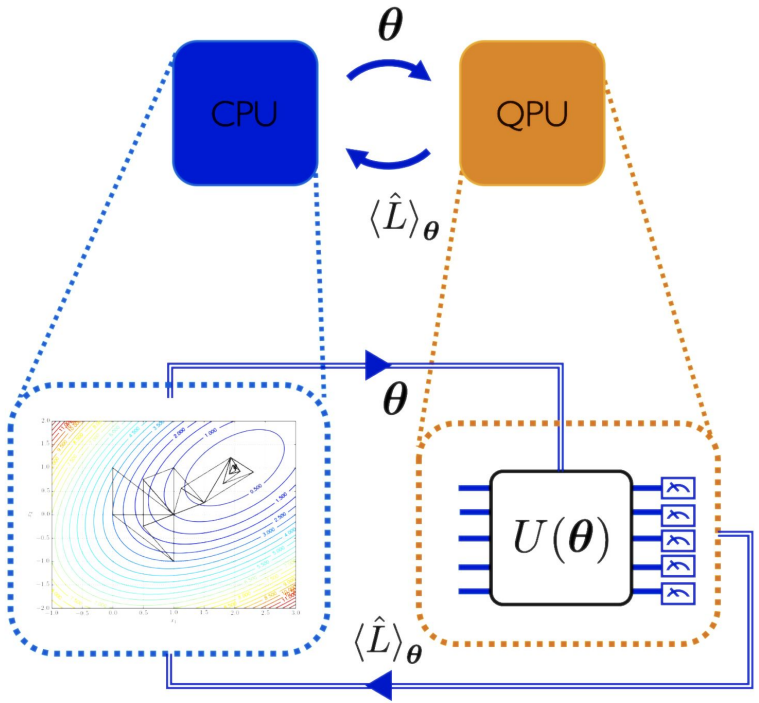
QML - Tensorflow Quantum

- Framework **híbrido** (quântico-clássico) baseado em **Tensorflow** e **Cirq**
- Auxilia na prototipação, treinamento, inferência e testes de modelos quânticos utilizando **quantum data**
- Fornece ferramentas para intercalar **algorítimos quânticos** e **lógica clássica**
- O treinamento é feito via a API **Keras**

Google Cloud

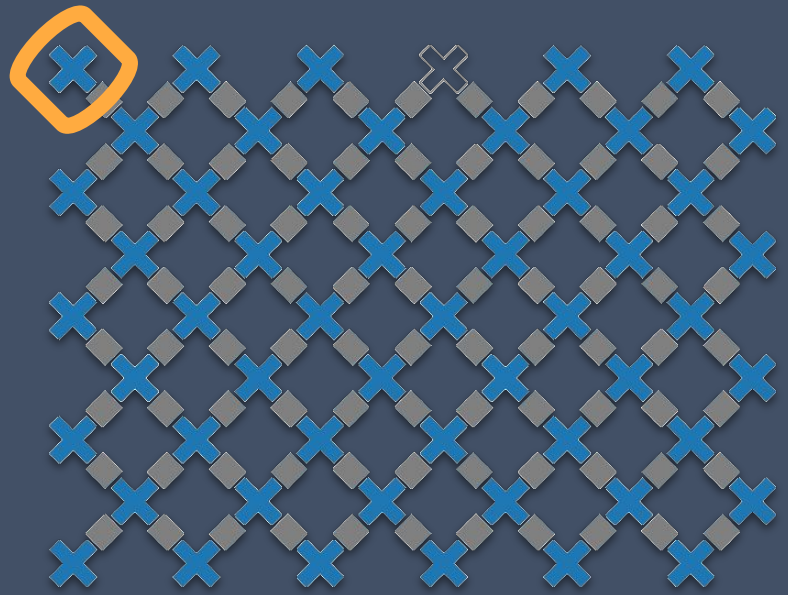
QML - Tensorflow Quantum
“Variational Quantum Algorithms”

- Executa o circuito quânticos na QPU
- Faz a medição do resultado <L> nas iterações
- Essa informação é disponibilizada à CPU
- O código de otimização sugere novos parms.

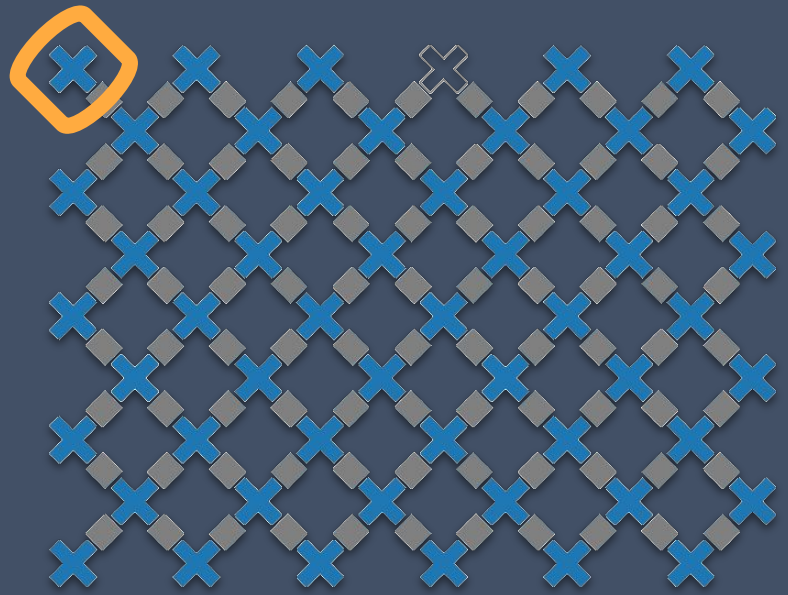


Google Cloud

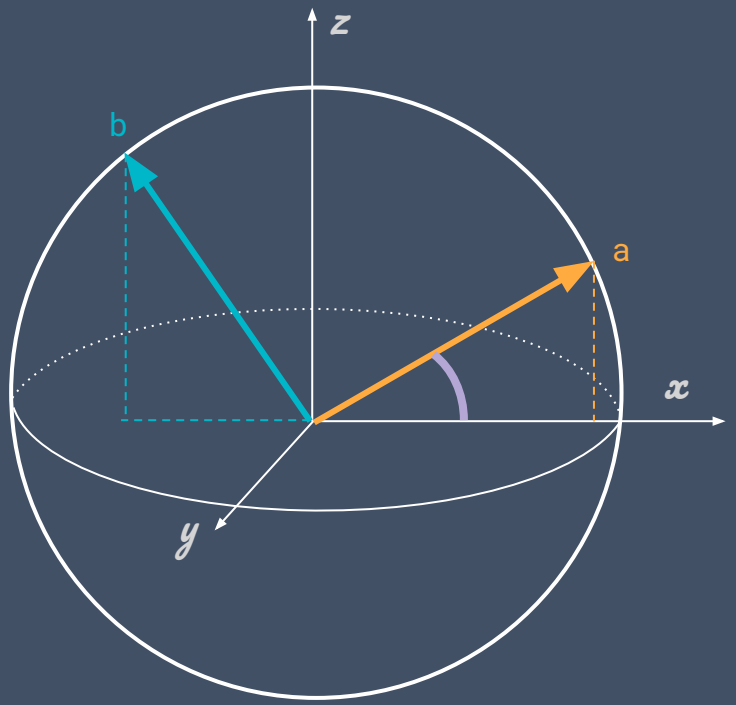
```
import cirq, random, sympy
import numpy as np
import tensorflow as tf
import tensorflow_quantum as tfq

qubit = cirq.GridQubit(0, 0)
```



```
import cirq, random, sympy
import numpy as np
import tensorflow as tf
import tensorflow_quantum as tfq

qubit = cirq.GridQubit(0, 0)

# Quantum data labels
expected_labels = np.array([[1, 0], [0, 1]])

# Random rotation of X and Z axes
angle = np.random.uniform(0, 2 * np.pi)
```



```
import cirq, random, sympy
import numpy as np
import tensorflow as tf
import tensorflow_quantum as tfq

qubit = cirq.GridQubit(0, 0)

# Quantum data labels
expected_labels = np.array([[1, 0], [0, 1]])

# Random rotation of X and Z axes
angle = np.random.uniform(0, 2 * np.pi)

# Build the quantum data

a = cirq.Circuit(cirq.Ry(angle)(qubit))
b = cirq.Circuit(cirq.Ry(angle + np.pi/2)(qubit))
quantum_data = tfq.convert_to_tensor([a, b])
```



q_data_input

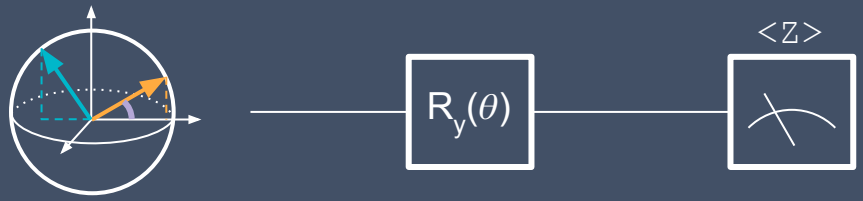
```
# Build the quantum model
q_data_input = tf.keras.Input(shape=(), dtype=tf.dtypes.string)
```

q_data_input

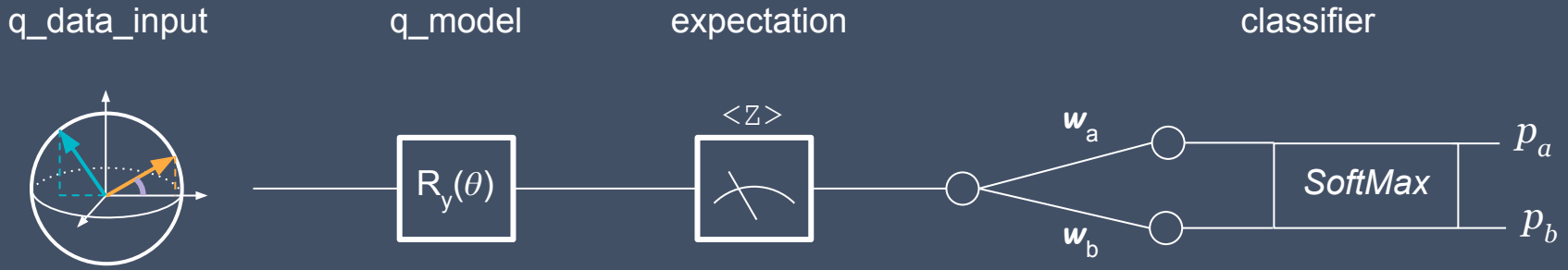
q_model


```
# Build the quantum model
q_data_input = tf.keras.Input(shape=(), dtype=tf.dtypes.string)

theta = sympy.Symbol('theta')
q_model = cirq.Circuit(cirq.Ry(theta)(qubit))
```

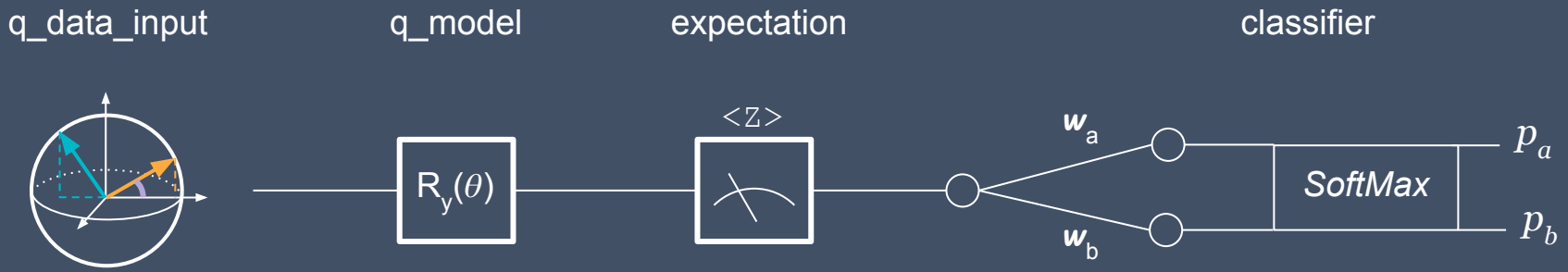
q_data_input q_model expectation



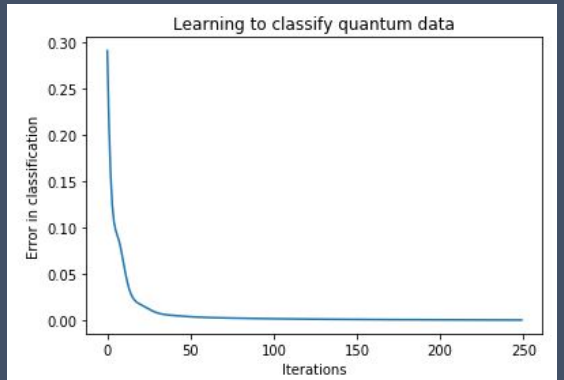

```
# Build the quantum model
q_data_input = tf.keras.Input(shape=(), dtype=tf.dtypes.string)

theta = sympy.Symbol('theta')
q_model = cirq.Circuit(cirq.Ry(theta)(qubit))

expectation = tfq.layers.PQC(q_model, cirq.Z(qubit))
expectation_output = expectation(q_data_input)
```




```
# Build the quantum model
q_data_input = tf.keras.Input(shape=(), dtype=tf.dtypes.string)

theta = sympy.Symbol('theta')
q_model = cirq.Circuit(cirq.Ry(theta)(qubit))

expectation = tfq.layers.PQC(q_model, cirq.Z(qubit))
expectation_output = expectation(q_data_input)

# Attach the classical SoftMax classifier
classifier = tf.keras.layers.Dense(2, activation=tf.keras.activations.softmax)
classifier_output = classifier(expectation_output)
```




```
# Train the hybrid model
model = tf.keras.Model(inputs=q_data_input,
     outputs=classifier_output)
model.compile(
     optimizer=tf.keras.optimizers.Adam(learning_rate=0.1),
     loss=tf.keras.losses.CategoricalCrossentropy())
history = model.fit(x=quantum_data, y=expected_labels,
     epochs=250, verbose=0)
```




```
Final loss value:
0.0007150595774874091
```

Demo Quântica

Google Cloud

Ecossistema - Quantum Computing & Quantum ML

QCWARE Forge

QuAIL qFlex

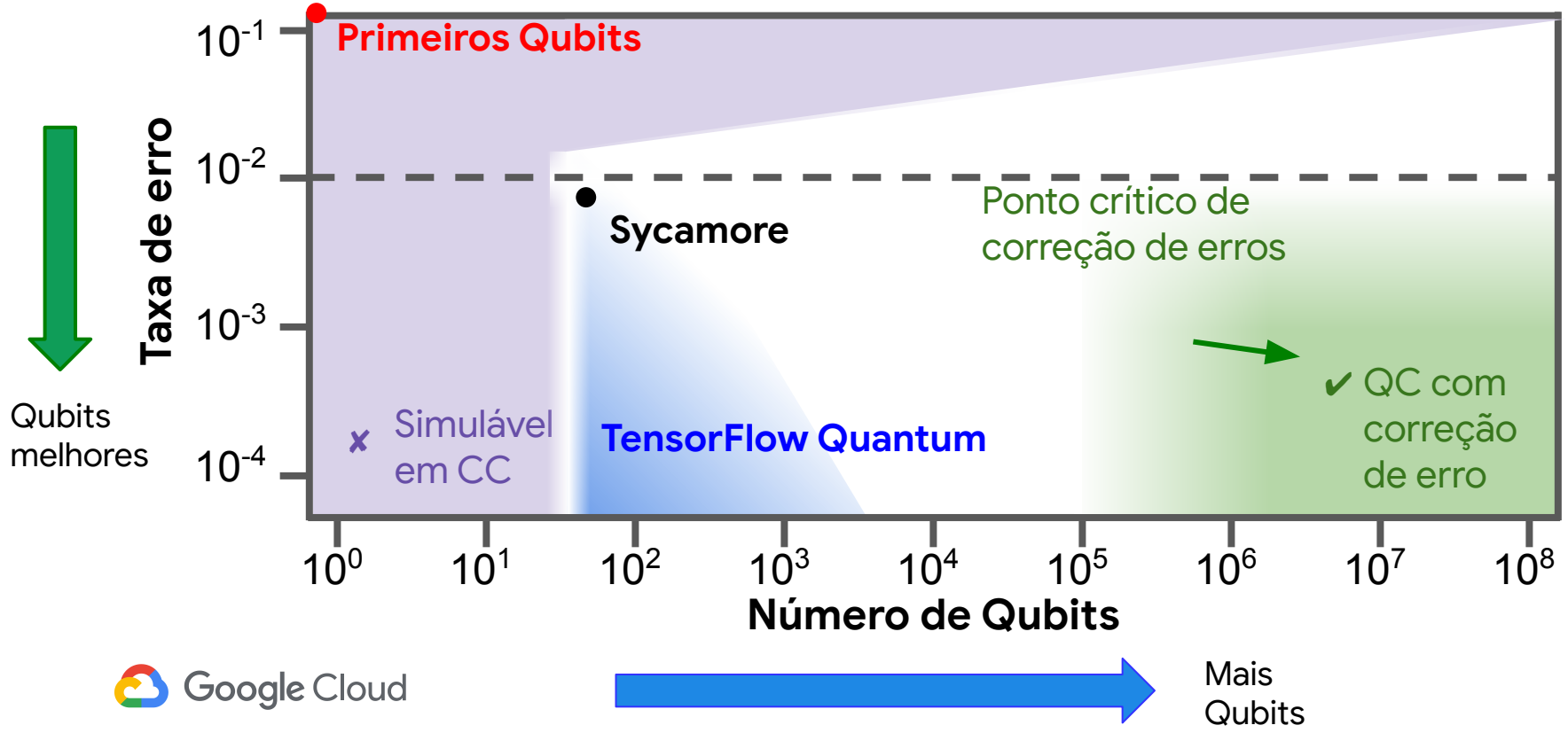
PENNYLANE

Google Cloud

Cambridge Quantum Computing

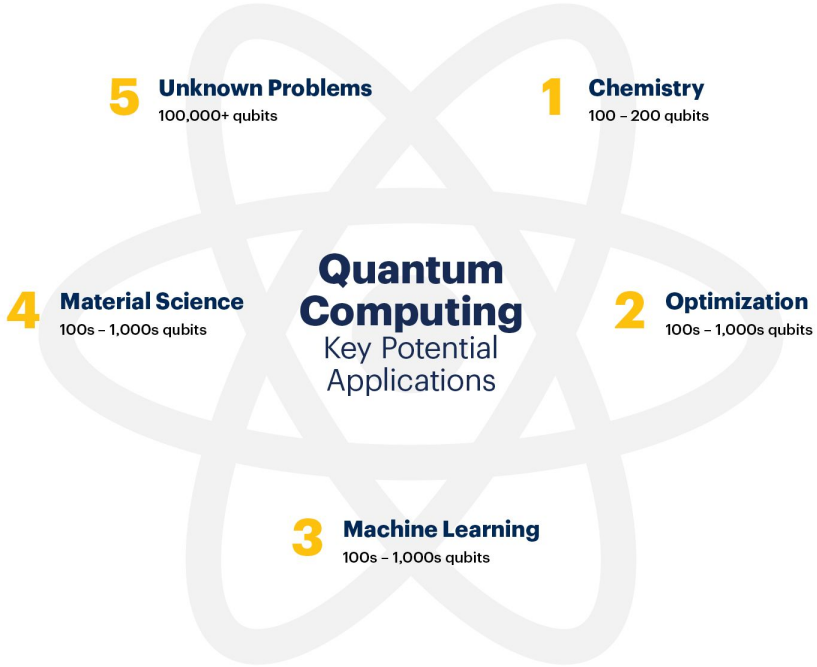
UCLA Center for Quantum Science & Engineering

Computação Quântica - Cenário Atual



Google Cloud

Computação Quântica - Cenário Atual



Google Cloud

gartner.com/SmarterWithGartner

Source: "Nature," Wikipedia

Gartner

Perguntas?

Google Cloud

Referências

- Google Research - Papers de Quantum Computing
- Google Research - Blog e Áreas de Pesquisa
- Google Cirq - Documentação
- Tensorflow Quantum - paper de criação do framework
- Tensorflow Quantum page
- TensorFlow Quantum: (TF Dev Summit '20)