# AQASM et myQLM, la solution quantique d'Atos

Devcon #18 - 100% Quantique - 23 février 2023 - ESGI Paris

PALO IT

Benoît Prieur (Palo IT) - CC0 - v1

# Objectifs de la présentation

- Présentation de la solution myQLM d’Atos
- Son langage AQASM
- Problème classique de la téléportation quantique
- Codage et simulation avec myQLM
- Usage de l'interopérabilité avec IBM Qiskit

# Rappel - états de Bell

Les quatres états de Bell d'intrication maximale

Les états de Bell

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B + |1\rangle_A \otimes |1\rangle_B) \quad (1)$$

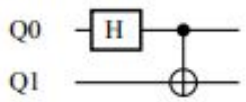
$$|\Phi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |0\rangle_B - |1\rangle_A \otimes |1\rangle_B) \quad (2)$$

$$|\Psi^+\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B + |1\rangle_A \otimes |0\rangle_B) \quad (3)$$

$$|\Psi^-\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B - |1\rangle_A \otimes |0\rangle_B) \quad (4)$$

# Simulation myQLM d'un état de Bell (n=10)

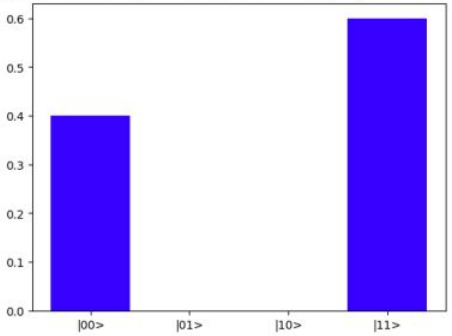
```python
from qat.lang.AQASM import Program, H, CNOT

qprog = Program()

nbqbits = 2
qbits = qprog.qalloc(nbqbits)

# Porte de Hadamard
qprog.apply(H, qbits[0])

# Porte CNOT
qprog.apply(CNOT, qbits[0], qbits[1])

circuit = qprog.to_circ()

# Affichage du circuit
# display(circuit) # Ne fonctionne pas sous Jupyter
%qatdisplay circuit --svg
```




Simulation avec n=10

```python
from qat.qpus import PyLinalg
import matplotlib.pyplot as plt

pylinalgqpu = PyLinalg()
job = circuit.to_job(nbshots=10)

result = pylinalgqpu.submit(job)

dictionnaire = {}
dictionnaire['|00>'] = float(0)
dictionnaire['|01>'] = float(0)
dictionnaire['|10>'] = float(0)
dictionnaire['|11>'] = float(0)

for sample in result:
    dictionnaire[str(sample.state)] = float(sample.p
    print("état quantique %s -> probabilité : %s" %

print(dictionnaire)
plt.bar(dictionnaire.keys(), dictionnaire.values(),
plt.show()
```

```
état quantique |11> -> probabilité : 0.6
état quantique |00> -> probabilité : 0.4
{'|00>': 0.4, '|01>': 0.0, '|10>': 0.0, '|11>': 0.6}
```



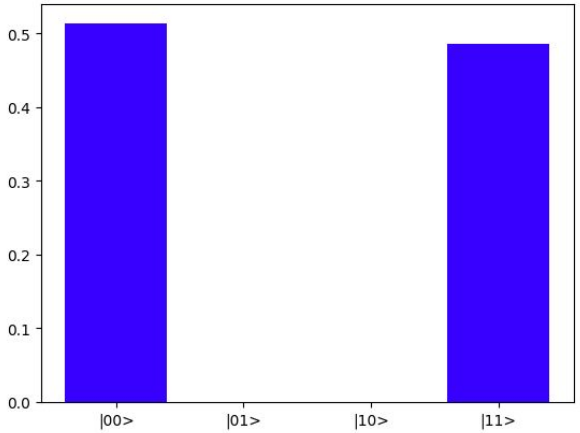
# Simulation myQLM d'un état de Bell (n=1000)

- Pour n=10, premier résultat statistique médiocre (0.4 et 0.6).
- On procède au même test avec n = 1000, résultat logiquement plus concluant (0.51 et 0.49)

Simulation avec n=1000

```
job = circuit.to_job(nbshots=1000)

result = pylinalgqpu.submit(job)

dictionnaire['|00>'] = float(0)
dictionnaire['|01>'] = float(0)
dictionnaire['|10>'] = float(0)
dictionnaire['|11>'] = float(0)

for sample in result:
    dictionnaire[str(sample.state)] = float(sample.probability)
    print("état quantique %s -> probabilité : %s" % (sample.state, sample.pro

print(dictionnaire)
plt.bar(dictionnaire.keys(), dictionnaire.values(), color='b')
plt.show()
```

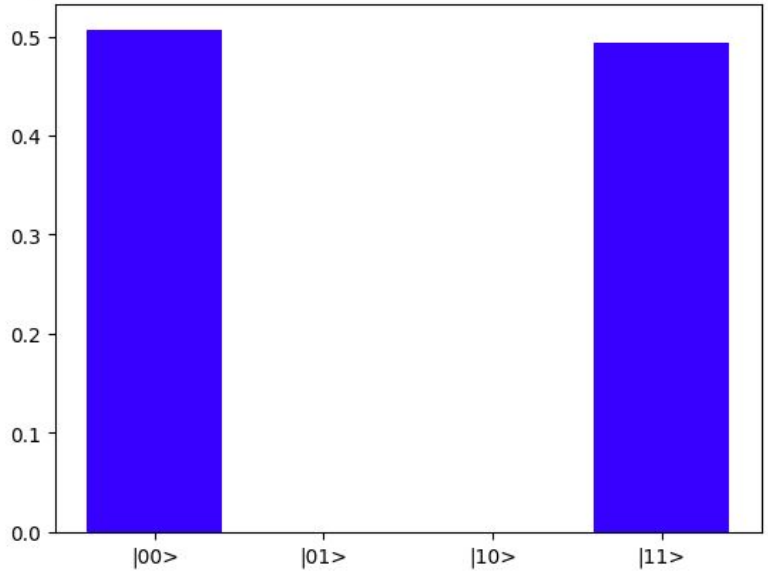
```
état quantique |00> -> probabilité : 0.514
état quantique |11> -> probabilité : 0.486
{'|00>': 0.514, '|01>': 0.0, '|10>': 0.0, '|11>': 0.486}
```



# Usage de l'interop. Qiskit sur la base du programme AQASM

```
# https://quantum-computing.ibm.com/
from qat.interop.qiskit import qlm_to_qiskit
from qat.interop.qiskit import BackendToQPU

job = circuit.to_job(nbshots=1000)
qiskit_circuit = qlm_to_qiskit(circuit)
MY_IBM_TOKEN = "*******"
qpu = BackendToQPU(token=MY_IBM_TOKEN, ibmq_backend="ibmq_qasm_simulator")

dictionnaire['|00>'] = float(0)
dictionnaire['|01>'] = float(0)
dictionnaire['|10>'] = float(0)
dictionnaire['|11>'] = float(0)

result = qpu.submit(job)

for sample in result:
    dictionnaire[str(sample.state)] = float(sample.probability)
    print("état quantique %s -> probabilité : %s" % (sample.state, sample.probability))

print(dictionnaire)
plt.bar(dictionnaire.keys(), dictionnaire.values(), color='b')
plt.show()
```

```
état quantique |00> -> probabilité : 0.507
état quantique |11> -> probabilité : 0.493
{'|00>': 0.507, '|01>': 0.0, '|10>': 0.0, '|11>': 0.493}
```

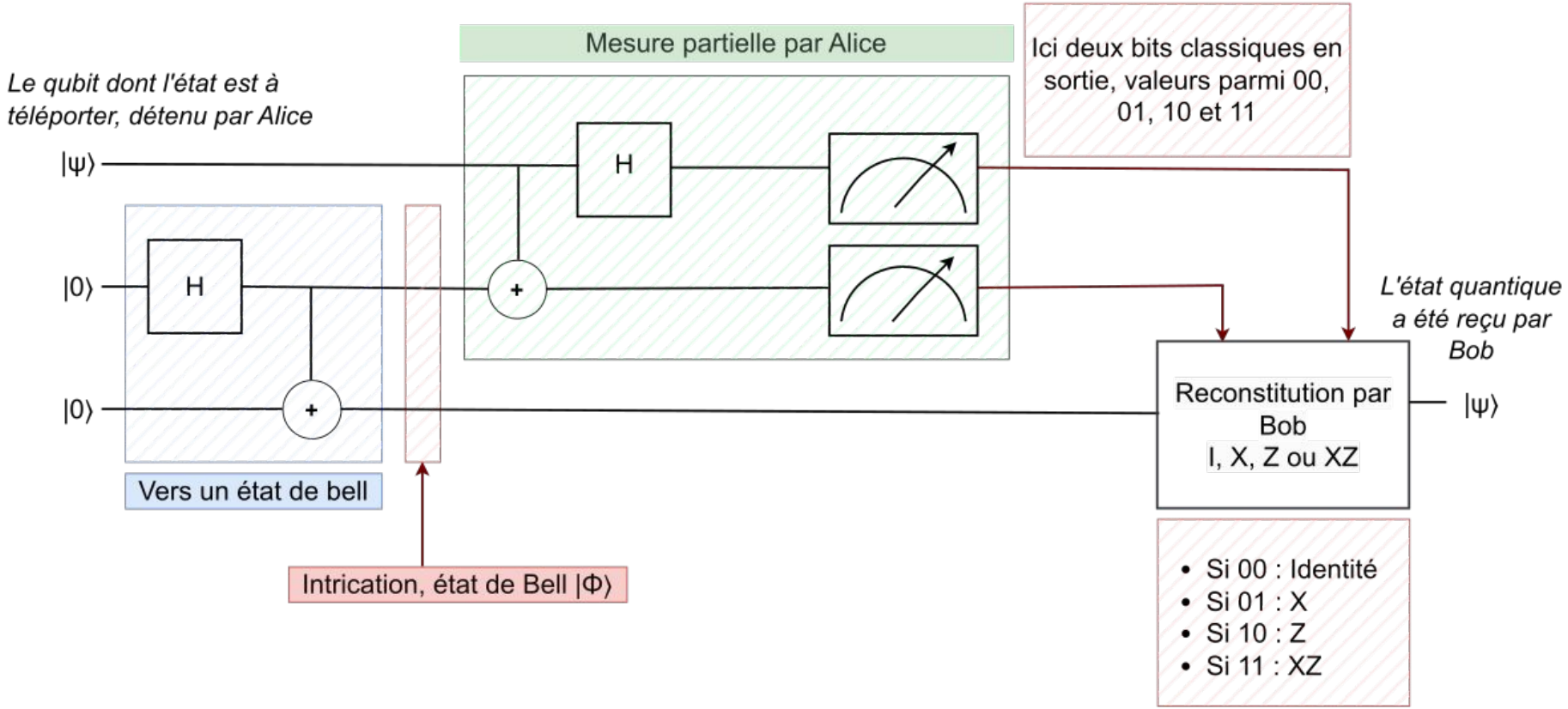



Usage du simulateur *ibmq_qasm_simulator*, dans le cloud.

# Téléportation quantique

- Copie d’une donnée transmise de Alice à Bob.
- Usage d’un état de Bell d’intrication maximale.
- Mesure partielle par Alice.
- Selon mesure classique, transformation Identité, X, Z, ou XZ.

# Téléportation quantique



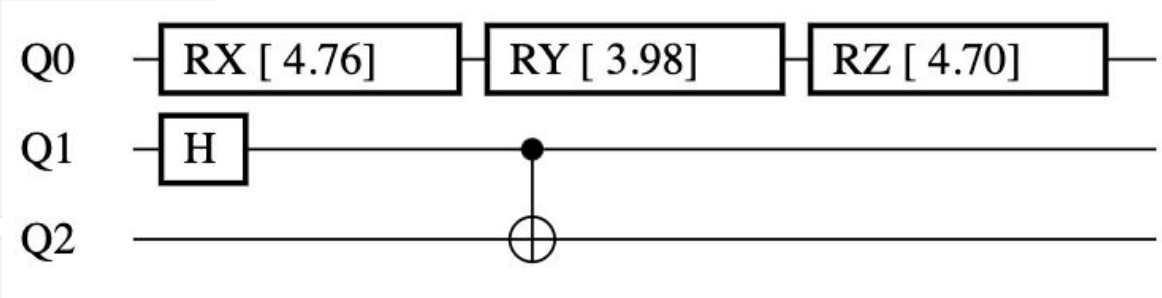

Benoît Prieur, CC0, via Wikimedia Commons

# Construction du programme de téléportation quantique - partie gauche

```
from qat.lang.AQASM import *

pr = Program()
qbit_teleporter_alice = pr.qalloc()
qbit_intrication = pr.qalloc()
qbit_bob = pr.qalloc()

bit_classique_intrication = pr.calloc()
bit_classique_teleportation = pr.calloc()

pr.apply(H, qbit_intrication)
pr.apply(CNOT, qbit_intrication, qbit_bob)
```

```
from random import random
from math import pi

pr.apply(RX(random() * 2 * pi), qbit_teleporter_alice)
pr.apply(RY(random() * 2 * pi), qbit_teleporter_alice)
pr.apply(RZ(random() * 2 * pi), qbit_teleporter_alice)

circ = pr.to_circ()
%qatdisplay circ --svg
```

1. Génération aléatoire de la valeur à transmettre.
2. Intrication quantique.



# Simulation AQASM (début)

```
import copy
pr2 = copy.deepcopy(pr)
```

```
from qat.qpus import PyLinalg

qpu = PyLinalg()

res = qpu.submit(circ.to_job())
for sample in res:
    print("état quantique %s -> probabilité : %s" % (sample.state, sample.probability))
```

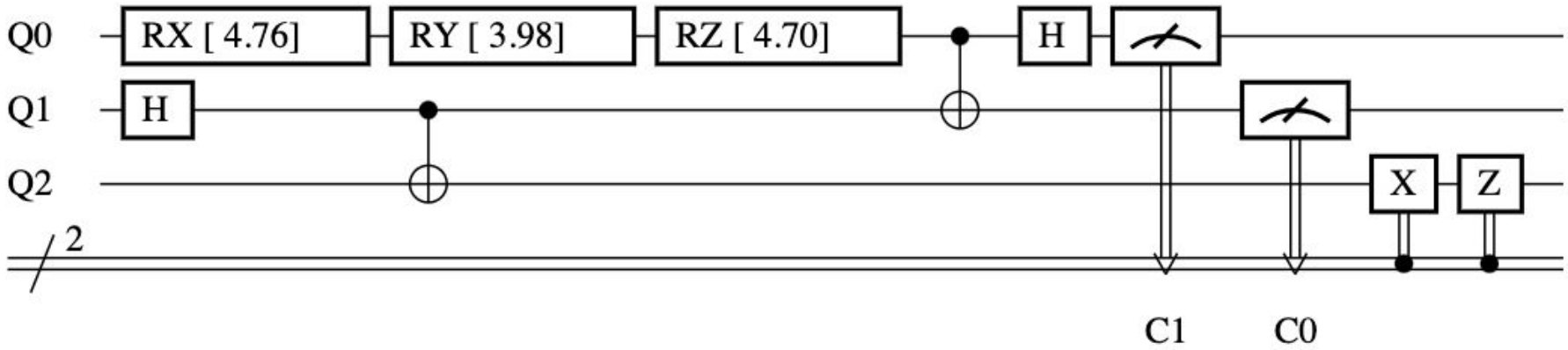
```
état quantique |0>|0>|0> -> probabilité : 0.24237143021741606
état quantique |0>|1>|1> -> probabilité : 0.24237143021741606
état quantique |1>|0>|0> -> probabilité : 0.25762856978258375
état quantique |1>|1>|1> -> probabilité : 0.25762856978258375
```

```
res = qpu.submit(circ.to_job(qubits=[qbit_teleporter_alice]))
for sample in res:
    print("état quantique %s -> probabilité : %s" % (sample.state, sample.probability))
```

```
état quantique |0> -> probabilité : 0.48474286043483206
état quantique |1> -> probabilité : 0.5152571395651675
```

# Simulation AQASM (suite) et code graphique

```
pr2.apply(CNOT, qbit_teleporter_alice, qbit_intrication)
pr2.apply(H, qbit_teleporter_alice)
pr2.measure(qbit_teleporter_alice, bit_classique_teleportation)
pr2.measure(qbit_intrication, bit_classique_intrication)

pr2.cc_apply(bit_classique_intrication, X, qbit_bob)
pr2.cc_apply(bit_classique_teleportation, Z, qbit_bob)

circ2 = pr2.to_circ()
%qatdisplay circ2 --svg
```



# Confirmation du résultat avec un simulateur Qiskit grâce à l'interop.

```
res2 = qpu.submit(circ2.to_job(qubits=[qbit_bob]))

for sample in res2:
    print("état quantique %s -> probabilité : %s" % (sample.state, sample.probability))
```

```
état quantique |0> -> probabilité : 0.48474286043483217
état quantique |1> -> probabilité : 0.5152571395651676
```

# Notebook Jupyter en ligne

Dépôt Github

- https://github.com/benprieur/Devcon-18-Quantique-Session-myQLM-AQASM/blob/main/Devcon%20%2318%20AQASM.ipynb