Informatique quantique

Institut Mines-Télécom Business School, webinaire, 29 septembre 2021

Benoît Prieur, Soarthec,

Sommaire de la présentation

1. Les raisons de l’engouement.
2. Les grands principes de l’informatique quantique.
3. La programmation quantique dans ses grandes lignes.
4. Un exemple concret : l’intrication maximale.
5. Conclusion. Vers un standard ?

Accélération de la couverture médiatique

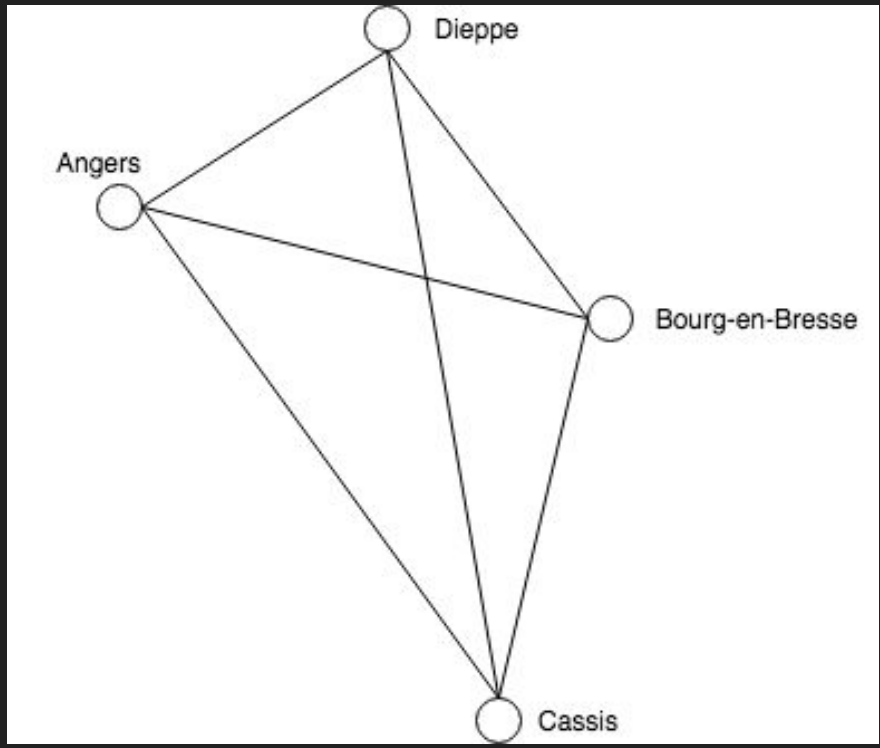
- Depuis quelques années.
- Généralement peu de contexte.
- Description du lien avec la physique quantique.
- Pas ou peu d’explication sur les avantages comparatifs.

Théorie de la complexité

- Force brute.
- Complexité des algorithmes : polynomiale (tout va bien) ou exponentielle (plus compliqué).
- Nombreux problèmes relatifs à l'industrie, le militaire, l'économie, le médical, l'apprentissage automatique, qui ne se résolvent pas en un temps polynomial.
- Heuristiques en recherche opérationnelle : obtention de solutions approchées.
- La conjecture du siècle : P =? NP.
- Un exemple de problème NP-Complet : le voyageur de commerce.

Le voyageur de commerce

- En notation de Landau : O(n!).
- n = 20 ; 1 microseconde par chemin.
- 115 jours pour obtenir la solution exacte.
- Approche théorique en nombre de qubits pour atteindre cette combinatoire : 55 qubits.



La dimension stratégique de l’informatique quantique

- Forte dimension géostratégique : implication des États, importance quasi-militaire.
- Communication de ce fait très régulée.
- Des annonces probablement en deçà des avancées les plus importantes.
- Dernière annonce d’ampleur : Google en 2018-2019 ; machine à 39 qubits.

Rappel sur les fondements

- L’infiniment petit : un monde de particules probabiliste et contre-intuitif.
- L’état quantique d’une particule.
  - Trois phénomènes fondateurs de l’informatique quantique :
    - Superposition.
    - Intrication.
    - Mesure quantique.

Petit détour par la notation bra-ket

Exemple à un qubit :

$$|z\rangle = a|0\rangle + b|1\rangle$$

Notation bra-ket : pile ou face

- “pièce quantique”.
- Mise en évidence de l’aspect probabiliste.
- | état quantique pièce 〉 = a | pile 〉+ b | face 〉
- Probabilité de quoi ?
  - De l’atteinte de l’état post-mesure.
- La mesure met fin à la superposition.

Le chat de Schrödinger

- Illustre deux notions : la superposition et la mesure.
- Explications.
- $| \text{chat} \rangle = a | \text{mort} \rangle + b | \text{vivant} \rangle$
  - Si équiprobable : $a = b = 1 / \sqrt{2}$
- Superposition comprise et démontrée dans les années 1920 et expérimentée en 1996 (Serge Haroche).

L’intrication quantique

- Même état quantique de deux particules, quelque soit la distance les séparant. On parle alors d’intrication ou d’enchevêtrement.
- Pas de partage d’information *a priori* ni quelconque forme de communication préalable.
- Controverse scientifique : Einstein/Bohr, « Dieu ne joue pas aux dés », le paradoxe EPR, voir aussi : inégalités de Bell (années 1960).
- Expérimentée en 1982 par Alain Aspect.

Système à deux qubits

- Approche vectorielle, à deux dimensions.
- On a comme valeurs cibles :
  - $| 00 \rangle$
  - $| 01 \rangle$
  - $| 10 \rangle$
  - $| 11 \rangle$

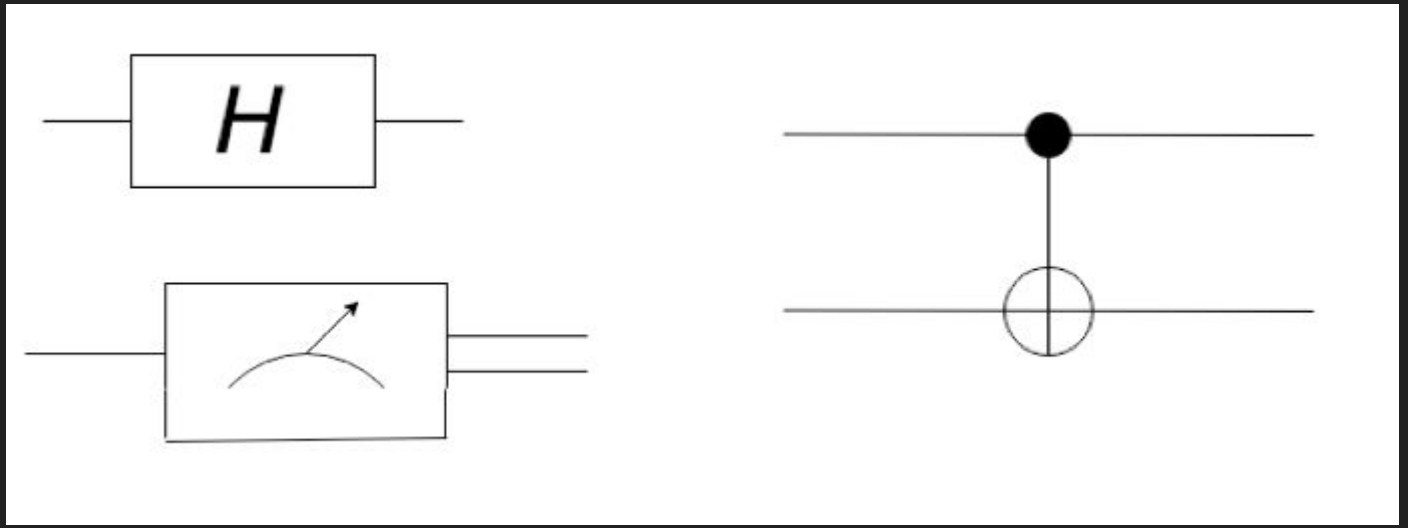
$| \text{état quantique} \rangle = a | 00 \rangle + b | 01 \rangle + c | 10 \rangle + d | 11 \rangle$

De la porte logique à la porte quantique

- Sur le modèle de la porte logique en informatique classique (algèbre de Boole, table de vérité). Transformation de valeurs de bits.
- Porte quantique : transformation matricielle des probabilités en entrée.

Quelques exemples de portes quantiques

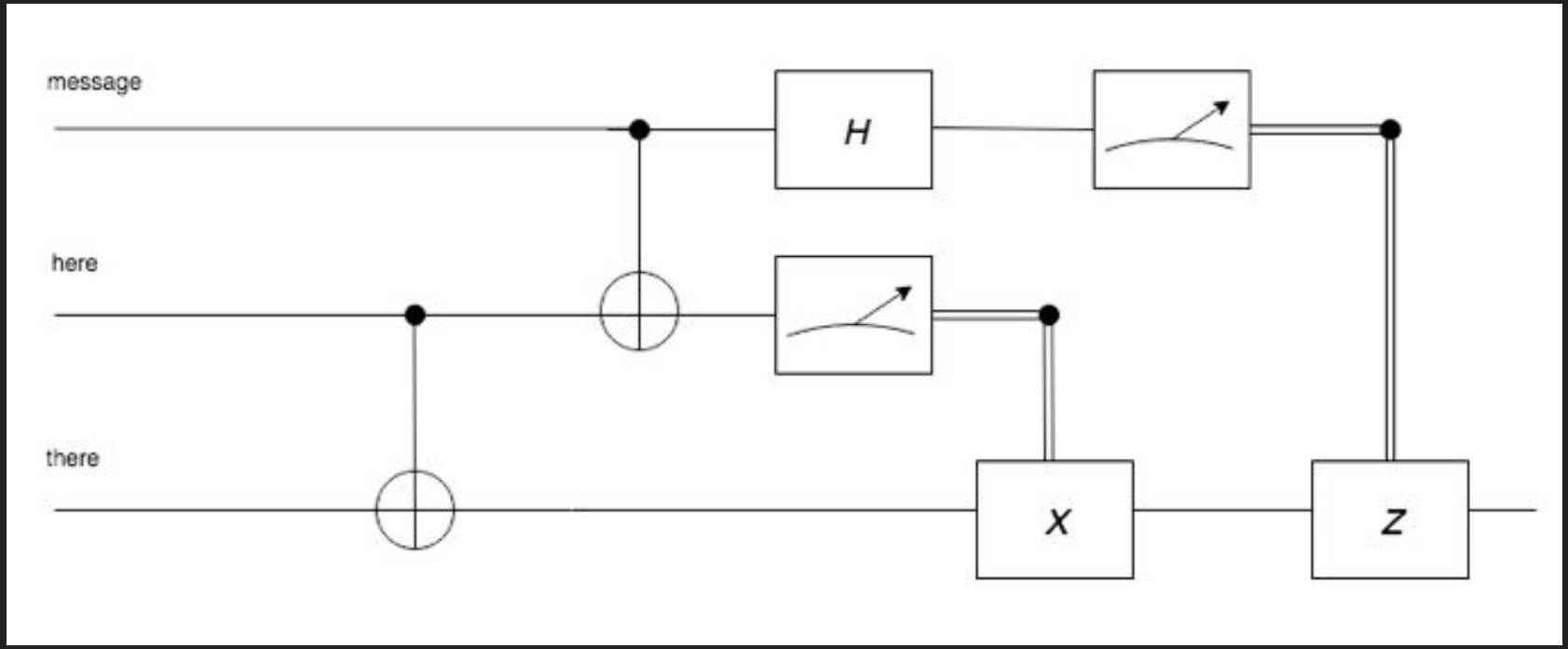
- Hadamard.
- CNOT.
- Mesure.



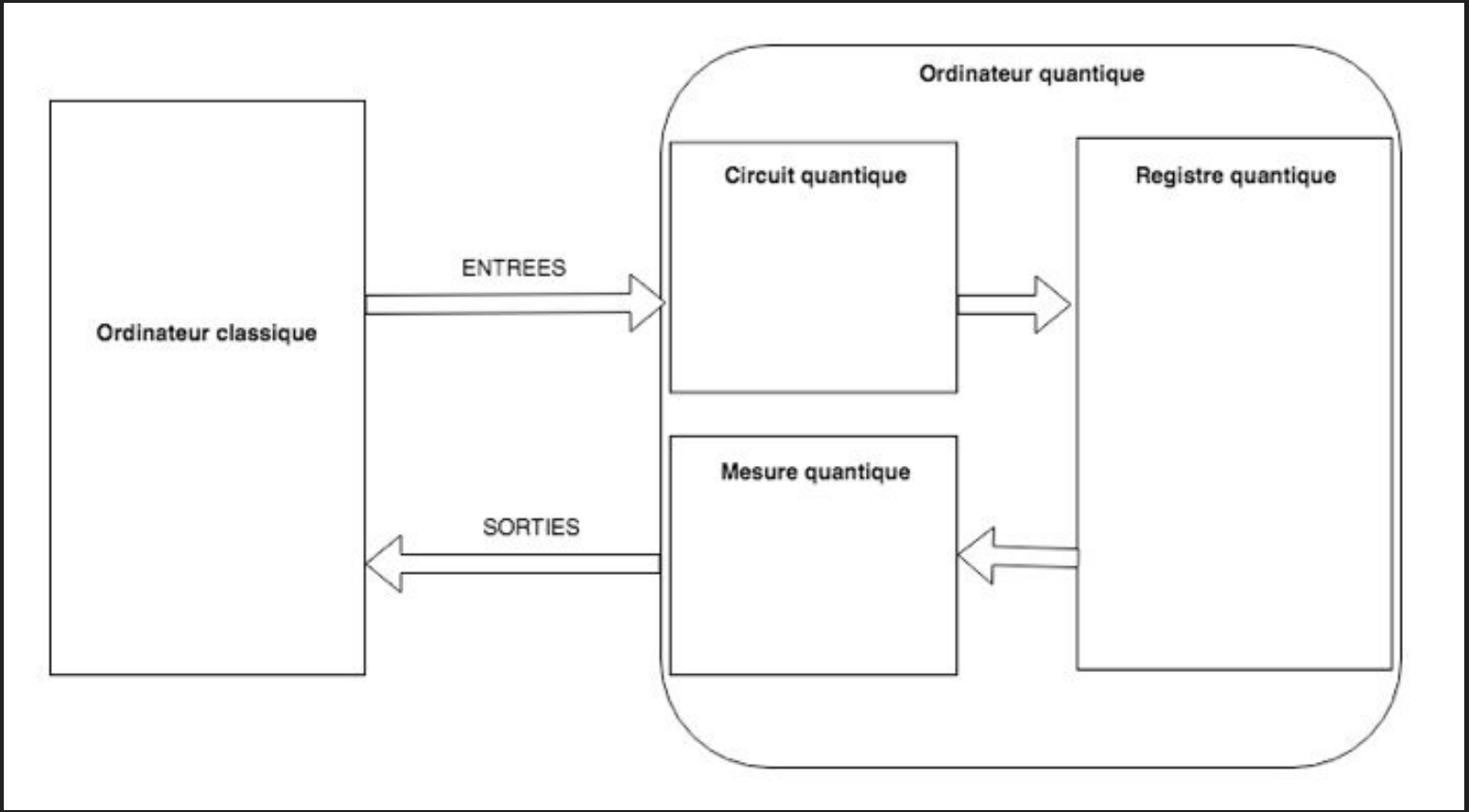
$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

$$\mathrm{CNOT} = cX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Exemple d’un circuit quantique



Le quantique au service de l'informatique classique



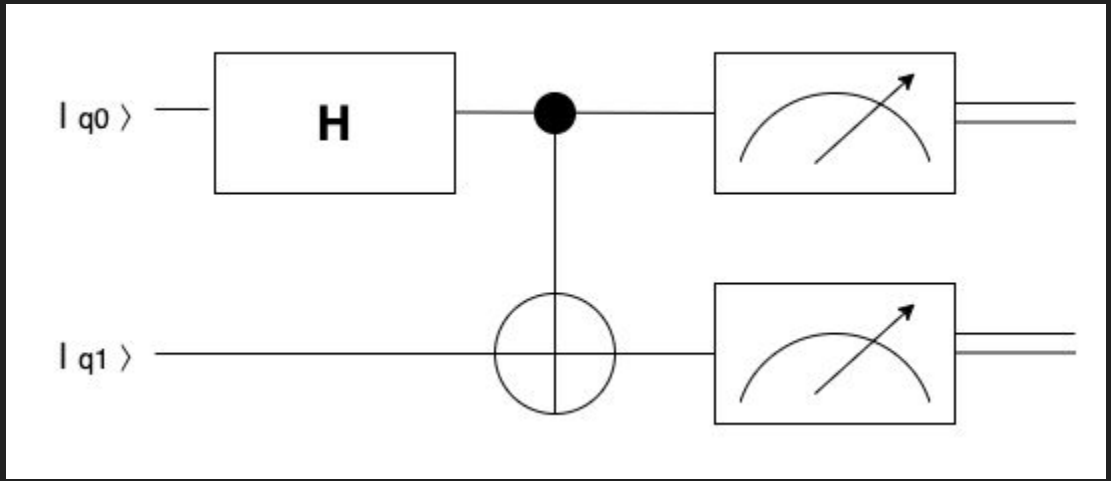
L’intrication maximale

Avec deux qubits. On cherche à obtenir les deux particules intriquées après mesure.

- En | 00 〉
- Ou en | 11 〉
- Non intriqués : | 01 〉
- Non intriqués : | 10 〉
- Les états en question sont nommés “états de Bell”.

Le circuit quantique de l’intrication maximale

- Juxtaposition d’une porte de Hadamard, d’une porte CNOT et de deux points de mesure.

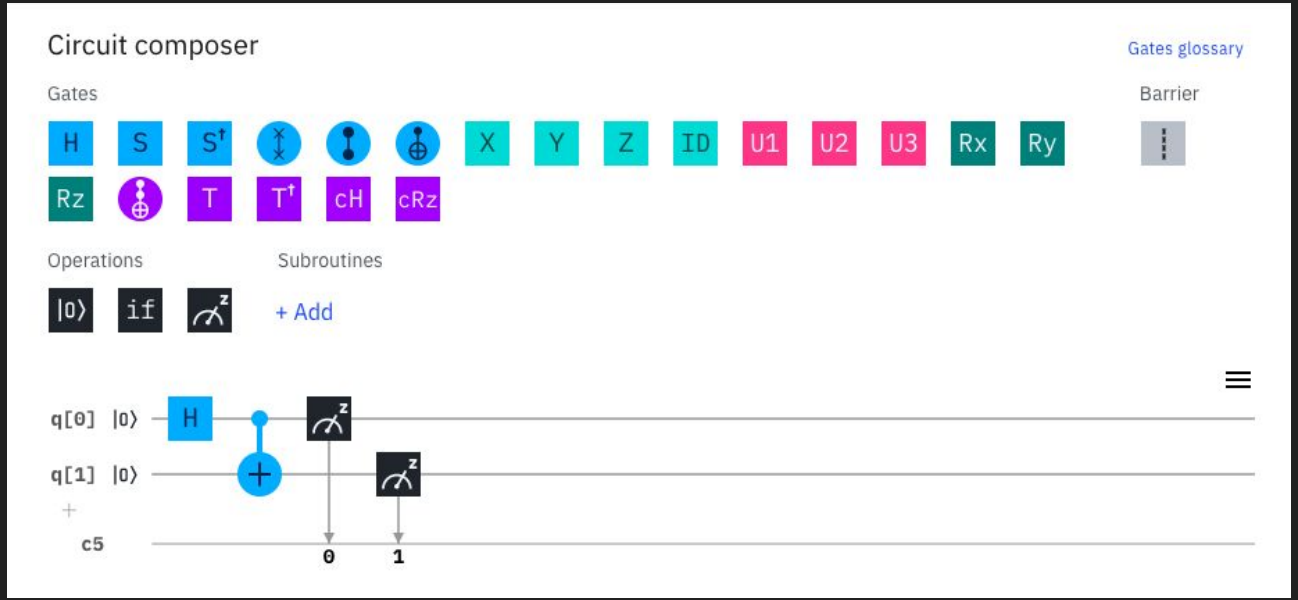


Principales plates-formes quantiques

- **Microsoft** Quantum, déclinaison cloud avec Azure.
- **IBM** Qiskit, incluant des simulateurs mais également de réelles machines quantiques accessibles.
- Solution myQLM par **Atos** avec un travail intéressant autour de l'interopérabilité.

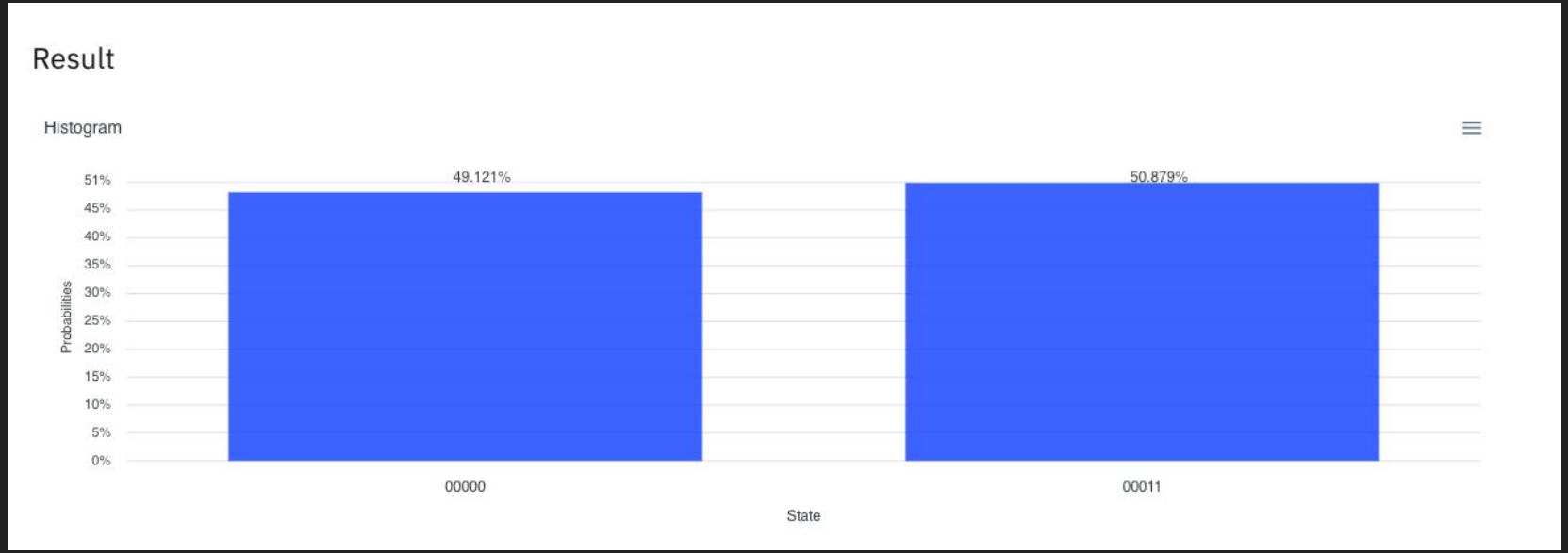
Implémentations diverses : IBM Qiskit

- Le *circuit composer*



Implémentations diverses : IBM Qiskit

- On sollicite 1000 fois le circuit quantique et on obtient ce résultat.



Implémentations diverses : IBM Qiskit

- On voit qu'on obtient 49,121 % des essais (503 essais) qui donnent le résultat suivant dans lequel les deux qubits de sortie sont identiques (à zéro) : | 00 〉
- On obtient également 50,879 % des essais (521 essais) qui donnent le résultat suivant dans lequel les deux qubits de sortie sont identiques (à un) : | 11 〉
- Pas de | 01 〉
- Pas de | 10 〉

Implémentations diverses : IBM Qiskit

- Test précédent avec un simulateur.
- Faisons le même test avec une véritable machine quantique.
- Profitons-en pour coder notre circuit quantique plutôt que d’utiliser le *composer*.
- Pour cela nous utiliserons un notebook *Jupyter* et le langage Python.

Implémentations diverses : IBM Qiskit

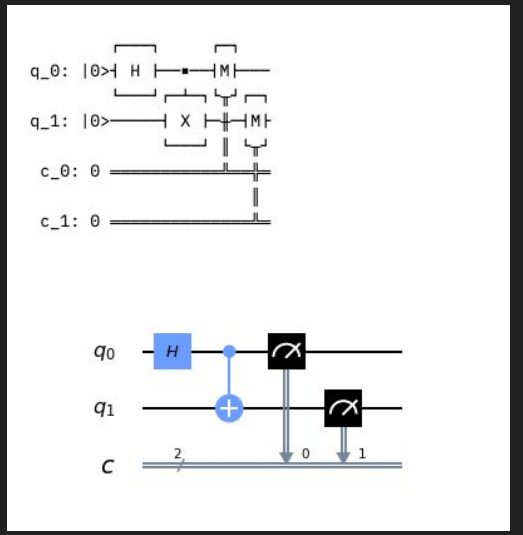
```
%matplotlib inline
import qiskit
from qiskit import(
  IBMQ,
  ClassicalRegister,
  QuantumCircuit,
  QuantumRegister,
  QuantumCircuit,
  execute,
  Aer)
import numpy as np
from qiskit.visualization import plot_histogram

circuit = QuantumCircuit(2, 2)
circuit.h(0)
circuit.cx(0, 1)
circuit.measure([0,1], [0,1])

# Circuit
circuit.draw()
print(circuit)
circuit.draw(output='mpl', filename='circuit.png')
```

circuit = QuantumCircuit(2, 2)

circuit.h(0)

circuit.cx(0, 1)

circuit.measure([0,1], [0,1])



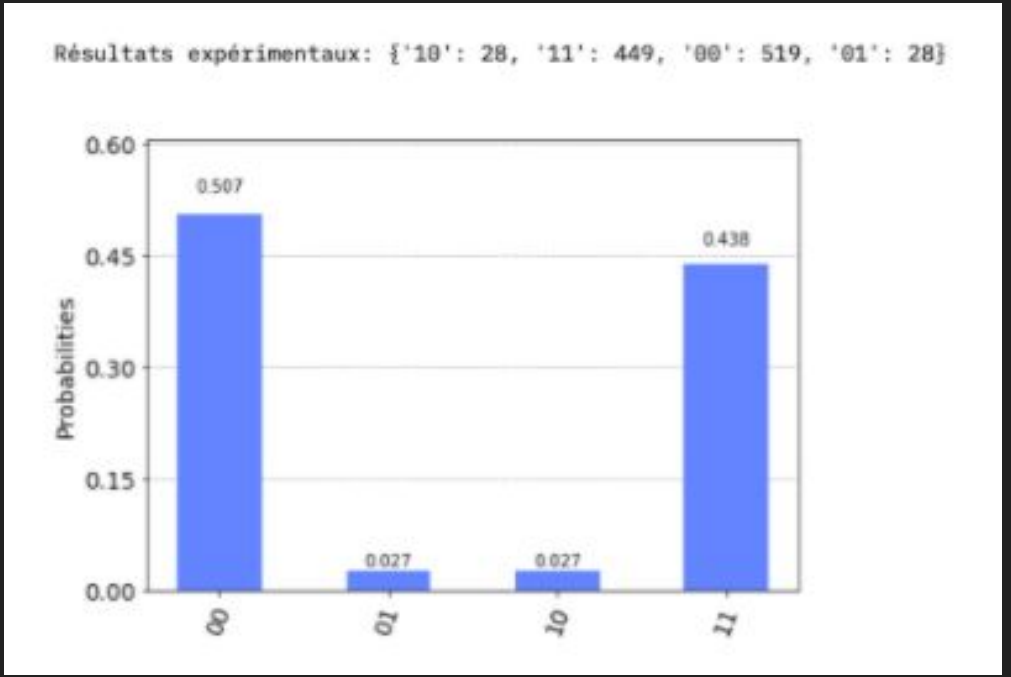
Implémentations diverses : IBM Qiskit

- On sollicite une réelle machine quantique IBM.

```
provider = IBMQ.get_provider(group='open')
device = provider.get_backend('ibmq_16_melbourne')
job_exp = execute(circuit, device, shots=1024)
result_exp = job_exp.result()
counts_exp = result_exp.get_counts(circuit)
print("\nRésultats expérimentaux:", counts_exp)
plot_histogram(counts_exp)
```

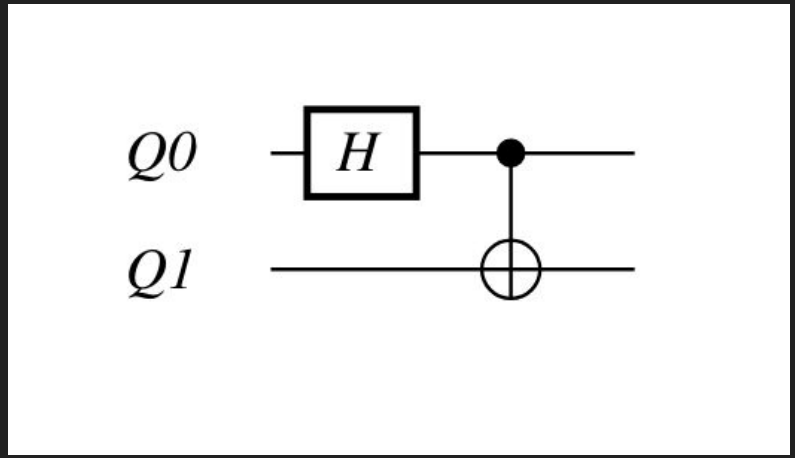
Implémentations diverses : IBM Qiskit

- Erreur quantique à 6 % environ.
- 6 % correspondant à
  - | 01 ⟩
  - | 10 ⟩



Implémentations diverses : Atos myQLM

```
from qat.lang.AQASM import Program, H, CNOT
qprog = Program()
nbqbits = 2
qbits = qprog.qalloc(nbqbits)
qprog.apply(H, qbits[0])
qprog.apply(CNOT, qbits[0], qbits[1])
circuit = qprog.to_circ()
%qatdisplay circuit
```



Implémentations diverses : Atos myQLM

- Gros point fort de myQLM, interopérabilité bidirectionnelle :
  - Circuit myQLM vers un solveur externe.
  - Circuit externe vers un solveur myQLM.

pip install myqlm-interop[qiskit_binder]

from qat.interop.qiskit import qlm_to_qiskit

from qat.interop.qiskit import BackendToQPU

Implémentations diverses : Atos myQLM

```
job = circuit.to_job(nbshots=333)

qiskit_circuit = qlm_to_qiskit(circuit)

MY_IBM_TOKEN = "xxxxxxxxxxxxxxxxxxx"

qpu = BackendToQPU(token=MY_IBM_TOKEN, ibmq_backend="ibmq_qasm_simulator")

result = qpu.submit(job)

for sample in result:

    print(sample.state)

    print(sample.probability)
```

Implémentations diverses : Atos myQLM

Résultat

|00> 0.5045045045045045

|11> 0.4954954954954955

Implémentations diverses : du côté de Microsoft

- Fort développement vers Azure.
- C# en pilotage, Q# côté quantique.
- Python comme alternative à Q#.

C#

```
using (var qsim = new QuantumSimulator())
{
    Result[] initials = new Result[] { Result.Zero, Result.One };
    foreach (Result initial in initials)
    {
      var res = BellTest.Run(qsim, 1000, initial).Result;
      var (numZeros, numOnes) = res;
    }
}
```

Q#

```
operation BellTest (count : Int, initial: Result) : (Int,Int)
{
    mutable numOnes = 0;
    using (qubits = Qubit[2])
    {
      for (test in 1..count)
      {
        Set (initial, qubits[0]);
        Set (Zero, qubits[1]);
        H(qubits[0]);
        CNOT(qubits[0],qubits[1]);
        let res = M (qubits[0]);
        if (res == One)
            set numOnes = numOnes + 1;
      }
      Set(Zero, qubits[0]);
      Set(Zero, qubits[1]);
    }
    return (count-numOnes, numOnes);
}
```

Conclusion. Vers un standard de développement ?

- L’émergence d’un langage ou méta-langage quantique autour de Python.
- Intérêt de l'interopérabilité.
- La sensibilité du sujet rend difficile la veille technologique.
- L’évolution des plate-formes est le point d’entrée judicieux de toute veille.
  - Ne pas privilégier une plate-forme en particulier.
  - D’autant qu’on observe un rapprochement des pratiques (à défaut d’une réelle standardisation).