EVIDEN

From NISQ to Hybrid Computing

With Q-Pragma

Océane KOSKA
PhD student & Engineer at Eviden
2026 - January, 15th and 16th

an atos business

Quantum – a new computing paradigm

Introduction

Quantum computing usefulness

Quantum computing promises exponential speed ups over classical computing for various tasks

Hybrid Computing era

use of QC in HPC applications

NISQ era

we are here

Improve Quantum Hardware

Understand what a hybrid quantum-classical application is

EVIDEN

NISQ and Hybrid Programming

NISQ vs HPQC

NISQ *(Noisy Intermediate Scale Quantum)* **era:**

- 10-100+ qubits
- noisy qubits
- short term

Demonstrate a quantum advantage – focus on the pure quantum part

- Prototyping programming languages *(e.g., Python)* – not HPC ones
- Short quantum computation *(to mitigate the impact of noise)*
- Low availability rate *(calibration, etc.)*

HPQC *(High Performance Quantum Computing)* **era:**

- ~thousands qubits
- ideal qubits (with QEC)

Exploit a quantum advantage in HPC applications

- HPC programming languages to accelerate existing application
- Long hybrid computation
- High availability rate

myQLM

Qiskit

EVIDEN

NISQ and Hybrid Programming

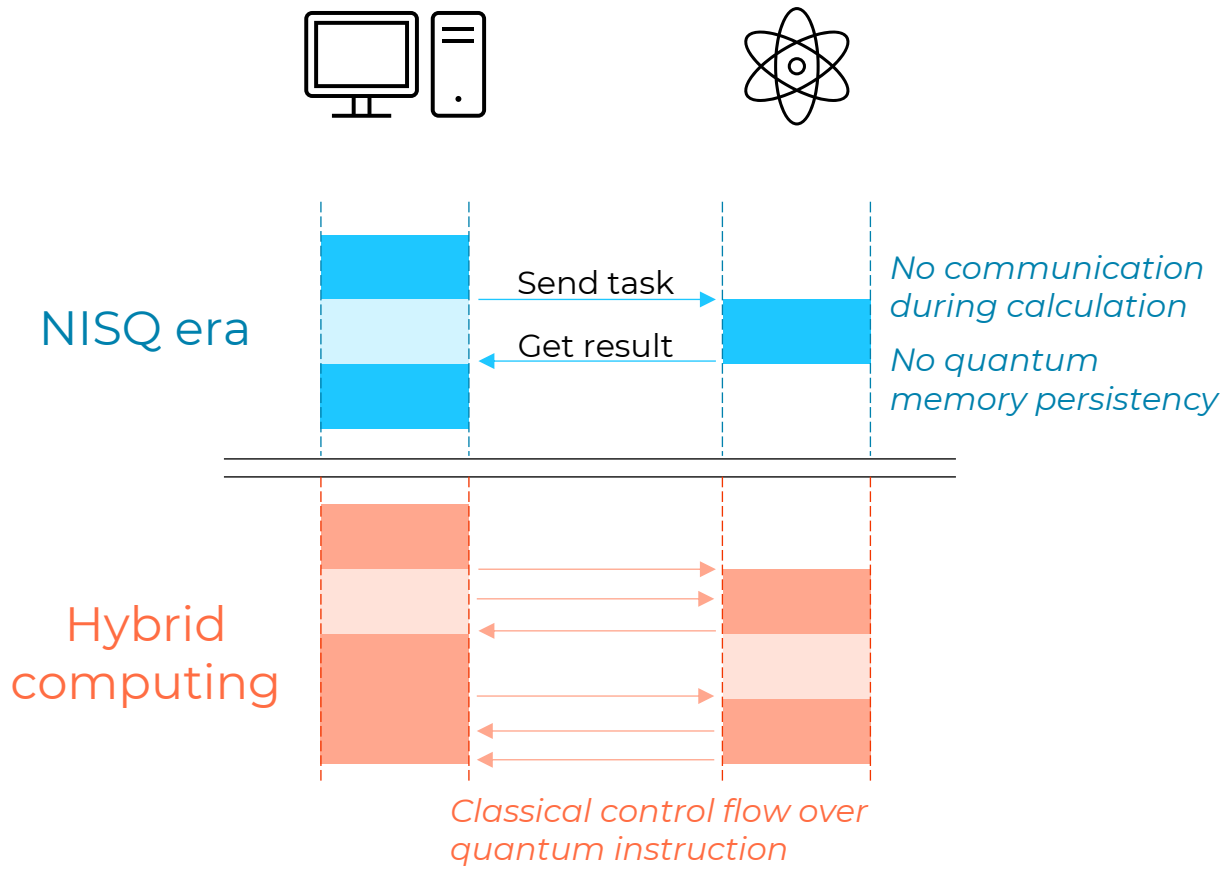
NISQ vs HPQC

NISQ *(Noisy Intermediate Scale Quantum)* **era:**

- 10-100+ qubits
- noisy qubits
- short term

HPQC *(High Performance Quantum Computing)* **era:**

- ~thousands qubits
- ideal qubits (with QEC)

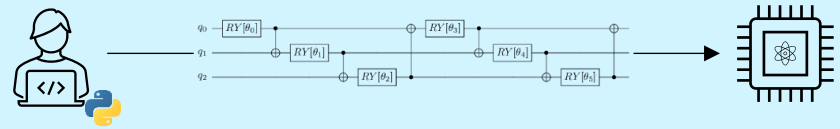


EVIDEN

NISQ Programming and Quantum Error Correction

Quantum Error Correction cannot be implemented using NISQ Frameworks

Circuit formalism *(used by most of QC programming framework)*

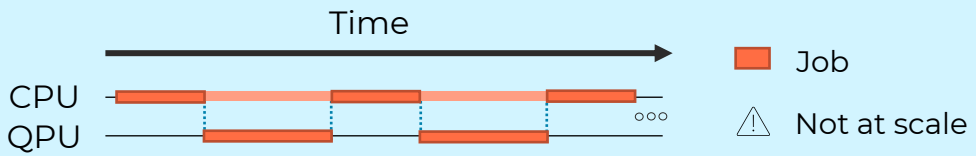


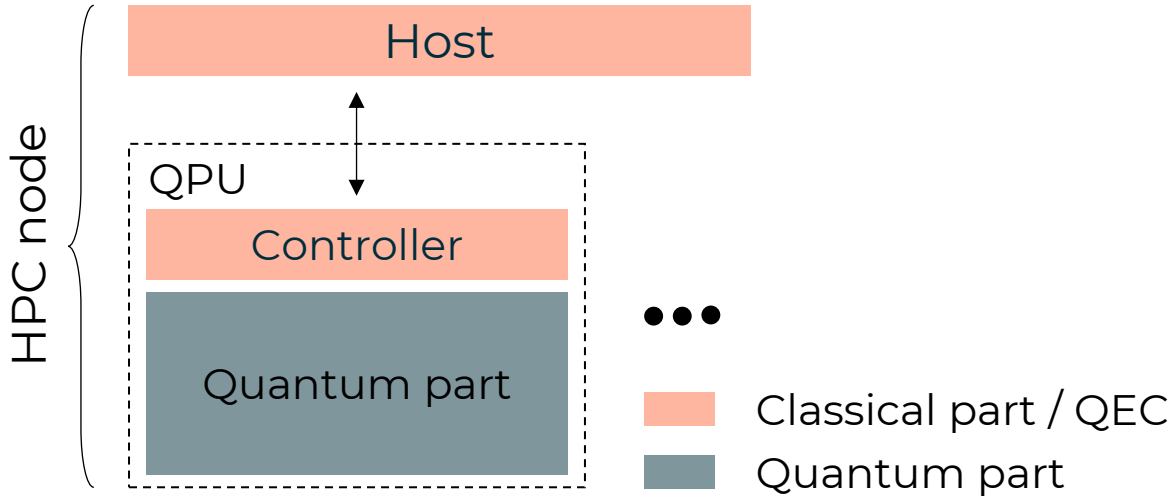
✓ User friendly

Fast to execute
(preloading)

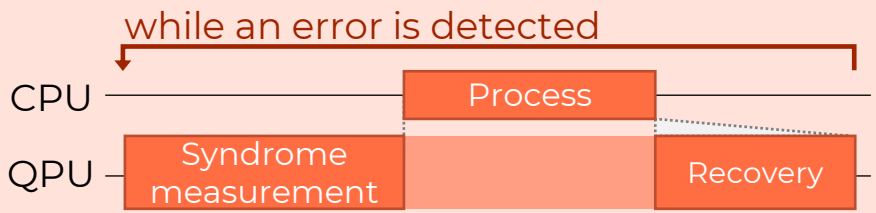
✗ Does not scale

Are static
(i.e. no interaction)





QEC



Windowed Quantum arithmetic

arXiv:1905.07682v1

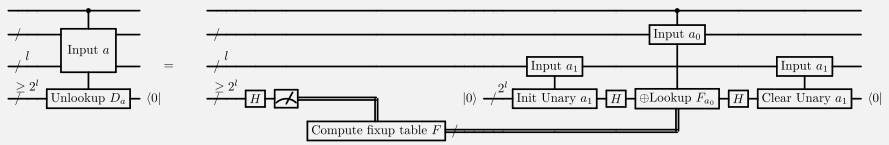
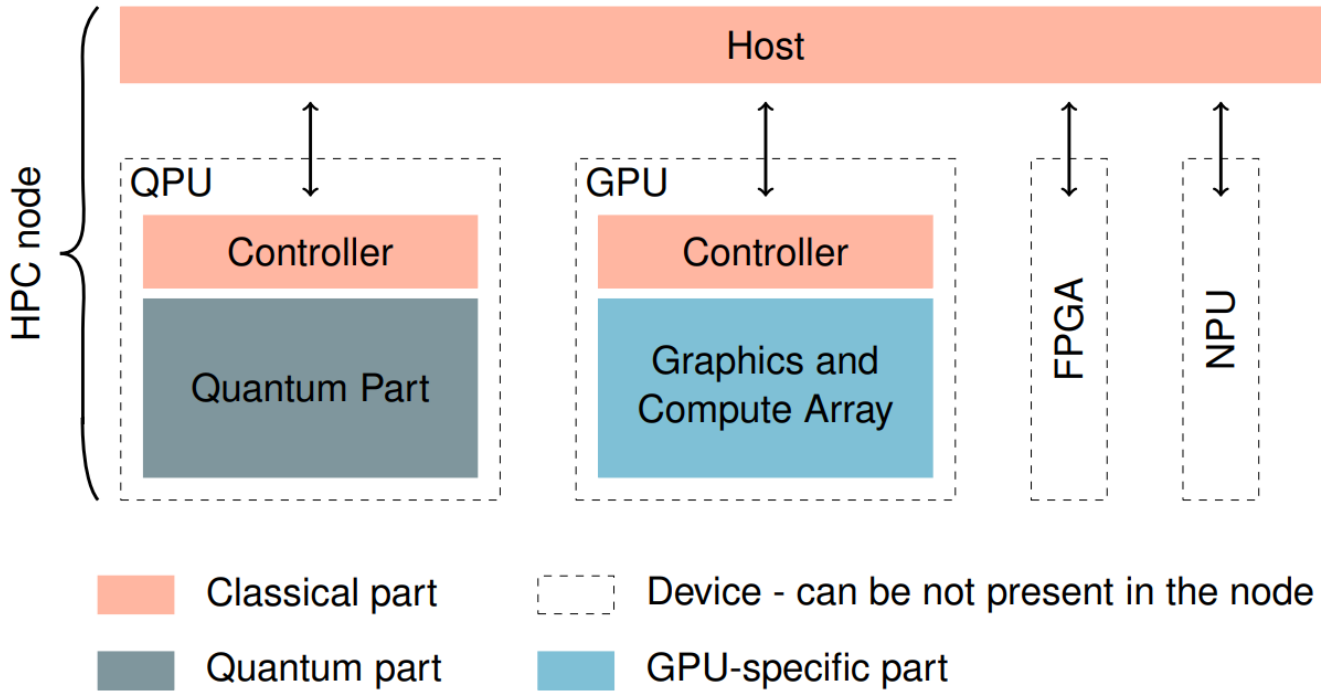


Figure 3: Uncomputing a table lookup with address space size L and an output size larger than $\sqrt{L}$. Has a Toffoli count and measurement depth of $O(\sqrt{L})$. Quadratically cheaper than computing the table lookup.

★ *used by Shor in 8 hours*

FTQC QPUs and HPC accelerators

Accelerators on a HPC node



GPU programming frameworks allow dynamic interaction

EVIDEN

Content Overview

01
Hybrid Computing & HPC

02
Q-Pragma – a C++ framework

03
Q-Pragma – overview of C++ pragmas

EVIDEN

01 Hybrid Computing & HPC

HPC hardware is already hybrid

Architecture of an hybrid CPU/GPU node

HPC is already hybrid. Some nodes are extended using GPUs

The Host is directly connected to the GPU using an HPC link *(like PCI express)*

Use of programming languages designed for performances *(e.g., C, C++, Fortran)*

Programming on GPUs

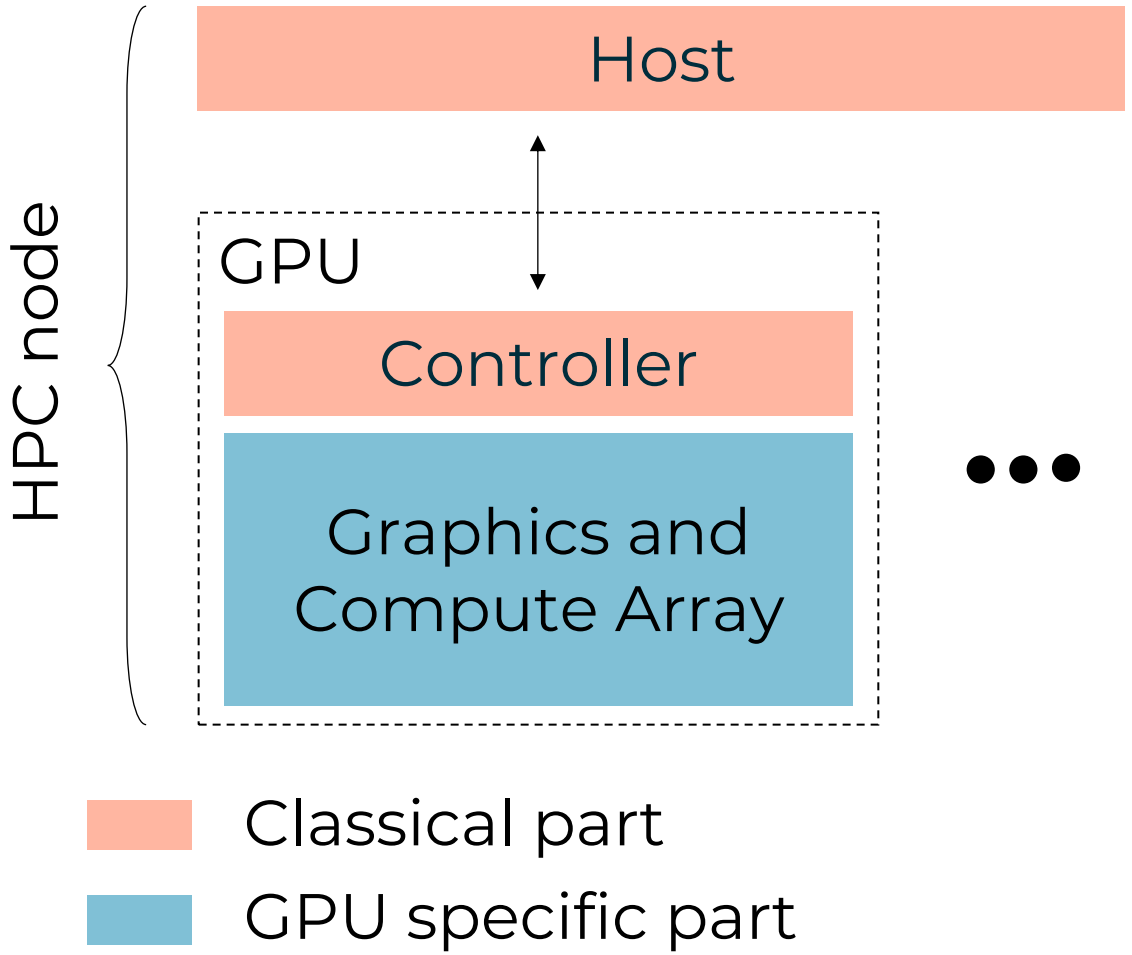
New language

OpenCL

NVIDIA CUDA

Language extension

OpenMP



Extending a programming language

Using pragmas

OpenMP

```
// Runs on CPU
uint8_t a, x[SZ], y[SZ];


// Runs on GPU
#pragma omp target map(to:x[0:SZ]) map(tofrom:y[0:SZ])
for (int i = 0; i < SZ; i++) {
    y[i] = a * x[i] + y[i];
}

// Runs on CPU
...
```

C++ & Quantum

```
// Runs on CPU
uint8_t a;
qpragma::quint_t<8UL> x, y;

// Runs on QPU
#pragma quantum scope
{
    y += a * x;
}

// Runs on CPU
...
```

Extending a programming language

Using pragmas

C++ & Quantum

Creating a superposition

$$|0\rangle + |1\rangle + \cdots + |200\rangle$$

```
// Create a uniform superposition [0:200]
quint8_t quantum_int;
qbool anc;
-------------------------------------------------------
do {
-------------------------------------------------------
    // Create a uniform superposition[0:255]
    reset(quantum_int);
    wall::H<8UL>(quantum_int);
-------------------------------------------------------
    #pragma quantum ctrl(quantum_int > 200)
    X(anc);
-------------------------------------------------------
} while (measure_and_reset(anc));
```

Create two quantum registers
One register containing the result & one ancilla

Reset first register *(by applying a measure)* and apply a wall of Hadamard gates to get the state $|0\rangle + |1\rangle + \cdots + |2^8 - 1\rangle$

Apply X gate is “quantum_int > 200”
$|0\rangle|0\rangle + |1\rangle|0\rangle + \cdots + |200\rangle|0\rangle + |201\rangle|1\rangle + \cdots + |255\rangle|1\rangle$

EVIDEN

02 Q-Pragma – a C++ framework

Extending C++ with two types of pragmas

Q-Pragma – A C++ framework

Pragmas for code locality

kernels and memory management

- *#pragma quantum scope*
- *#pragma quantum move*

```
#pragma quantum scope
{
    // Code executed on the QPU
    ...

    // Move a var from Host to Device
    #pragma quantum move toDevice(variable)
    ...
}
```

Pragmas for quantum computing

exploits the properties of quantum hardware

- *#pragma quantum routine*
- *#pragma quantum ctrl*
- *#pragma quantum else*
- *#pragma quantum compute*

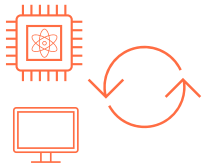
ⓘ Exploit the capabilities of the quantum device
reversibility, controllability, etc.

? What are the capabilities of a Quantum Programming Framework?

Quantum capabilities

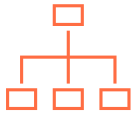
Q-Pragma – a C++ framework

Code and memory locality

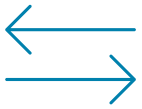


Dynamic interaction

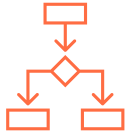
Scalability



Typing



Reversibility



Controllability

Safe uncomputation

Quantum capabilities are not supported by existing frameworks

Q-Pragma – a C++ framework

✓ Fully supported
∼ Partially supported

Framework	Locality	Dynamic interaction	Scalability	Typing	Reversibility	Controllability	Safe uncomputation
Cirq [8]					✓	✓	
CUDA Quantum [11]	∼				✓	✓	∼
myQLM [13]				✓	✓	✓	∼
OpenQASM 3 [9]		✓	✓		✓	✓	
ProjectQ [33, 18]		✓	✓		✓	✓	∼
Q# [36]		✓	∼		✓	✓	∼
QCOR [26]	∼				✓	✓	
Qiskit [31]					✓	✓	
Quipper [17]					✓	✓	✓
Scaffold [1]						✓	
Silq [5]				✓	✓	✓	✓
CUDA [30]	✓	✓	✓	✓			
OpenCL [34]	✓	✓	✓	✓			
OpenMP [10, 24]	✓	✓	✓	✓			

Table 1: Features implemented by existing hybrid frameworks - the first group corresponds to quantum frameworks, the second one corresponds to classical frameworks (see Appendix B for details).

Q-Pragma – Quantum types and C++ pragma directives

Q-Pragma – a C++ framework

Quantum types

- Quantum bool (or qubit) — `qbool`
- Quantum integers — `quint8_t, qint8_t, …, quint_t<SIZE>, qint_t<SIZE>`
- Array of qubits — `qpragma::array<SIZE>`
- …

Quantum pragmas

- Quantum scope — `#pragma quantum scope [with (…)]`
- Quantum move — `#pragma quantum move [toDevice(…)] [toHost(…)]`
- Quantum routine — `#pragma quantum routine [params]`
- Quantum compute — `#pragma quantum compute`
- Quantum ctrl — `#pragma quantum ctrl (…)`
- Quantum else — `#pragma quantum else`

EVIDEN

03 Q-Pragma – Overview of C++ pragmas

Q-Pragma – Creating advanced quantum kernels

Example of *quantum routine* directive

Code example: routine directive

```
#pragma quantum routine
void bell_pair(const qbool & qb0,
               const qbool & qb1) {
    H(qb0);
    CNOT(qb0, qb1);
}
```

Definition of a pure quantum kernel.

This kernel is:

- Callable *(as any function)*
- Reversible
- Controllable

```
int main() {
    …;
    ::bell_pair(qb0, qb1);
    ::bell_pair.dag(qb0, qb1);
    ::bell_pair.ctrl(qc, qb0, qb1);
}
```

Call different version of *bell_pair* quantum kernel

Q-Pragma – Controlling instructions with a qubit

Example of *quantum ctrl* directive

Code example: ctrl directive

```
// Create a uniform superposition [0:200]
quint8_t quantum_int;
qbool anc;

do {
    // Create a uniform superposition[0:255]
    reset(quantum_int);
    wall::H<8UL>(quantum_int);

    #pragma quantum ctrl(quantum_int > 200)
    X(anc);
} while (measure_and_reset(anc));
```

Apply X gate if *"quantum_int > 200"*

Q-Pragma – Safe uncomputation

Example of *quantum compute* directive

Code example: compute directive

```
#pragma quantum routine (double angle)
void RZ_2qb(const qbool & qb1, const qbool & qb2) {
    // Define qbool
    qbool ancilla;
    {
        #pragma quantum compute
        { CCNOT(qb1, qb2, ancilla); }

        // Apply RZ gate
        (RZ(angle))(ancilla);

        // Hidden uncompute
    }
}
```

Create a parametrized routine

Compute "*ancilla*" register

Uncompute automatically at the end of the scope

Reminder

A quantum routine is a pure quantum function.
Then, a routine is:

- *Callable* ***(as any function)***
- *Reversible*
- *Controllable*

Q-Pragma – an advanced standard library

Shor algorithm, using Q-Pragma arithmetic operators

Code example Shor algorithm in 10 lines

```
uint64_t to_divide = …, random_base = …, measurement = 0UL;

#pragma quantum scope with (to_divide, random_base, measurement)
{
    qint_t<QSIZE> first_register;
    wall::H<QSIZE>(first_register);

    qint_t<QSIZE> second_register =
        qpragma::pow(random_base, first_register) % to_divide;
    reset(second_register);

    qft<QSIZE>(first_register);
    measurement = measure_and_reset(first_register);
}
```

EVIDEN

Thank you

For more information, please scan:

Papier published in
Science of Computer Programming

Contacts: oceane.koska@eviden.com arnaud.gazda@eviden.com