PyCOMPSs for CUNQA

Galicia Supercomputing Center (CESGA)

Marta Rilo Martínez



"This work has been financially supported by the Ministry of Economic Affairs and Digital Transformation of the Spanish Government through the QUANTUM ENIA project call - Quantum Spain project, and by the European Union through the Recovery, Transformation and Resilience Plan - NextGenerationEU within the framework of the Digital Spain 2026 Agenda".

CESGA
GALICIA SUPERCOMPUTING CENTER

COMPSs and PyCOMPSs



COMPSs (BSC)

Task-based programming model. Parallel code for distributed systems.

MULTI-LANGUAJE

Java core, with support for Python (**PyCOMPSs**), C/C++ and R

VERSION

PyCOMPSs v3.4 “Poinsettia”

```
@constraint(processors=[{'processorType':'GPU',...}])   1
@task(returns=1)                                        2

def func(a, b, c):
    ...
    return result
```

Code adapted from the COMPSs 3.4 documentation (@constraint decorator)

1. **@constraint**
   Defines which processor type executes the task
2. **@task**
   Indicates that the function must run in parallel

Limitation: PyCOMPSs only recognized classical processors

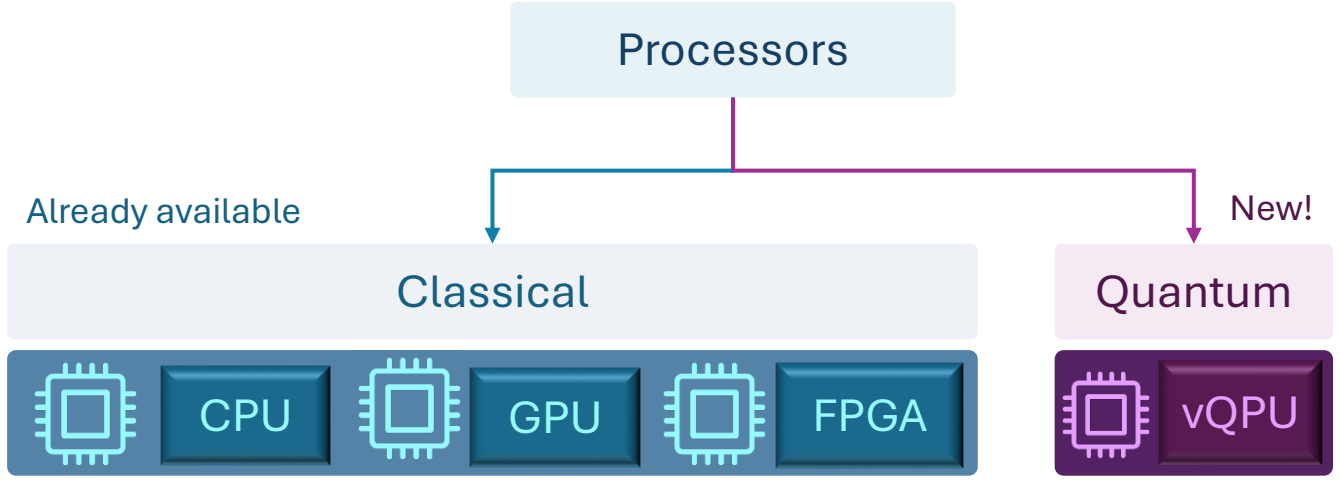
From Classical to Hybrid Model

MOTIVATION

- COMPSs supports only **classical processors**
- **Qdislib** (quantum, circuit cutting) with no direct QPU support

OBJECTIVE

- To integrate **QPUs**
- Enable unified **hybrid workflow**



Qmio Infrastructure and COMPSs Installation

The infrastructure selected was Qmio from CESGA



Frontal	
Cores	64
Mem	1TB
IB	Yes

x86	COMPSs installed
Cores	64
Mem	1TB
IB	Yes

a64	
Cores	48
Mem	32GB
IB	Yes

QPU	
Qubits	32
Mem	-
IB	No



Installation in Supercomputers

Supercomputers/

qmio_x86.cfg

Characteristics of the system
(memory of the nodes, cores per node, interconnection network, etc.)

Queues/

slurm_qmio.cfg

Dictionary between COMPSs and SLURM configuration in Qmio

Classical Benchmark with KMeans

To check if COMPSs was working correctly on Qmio, we performed a benchmark

1 BENCHMARK

KMeans

Classical clustering algorithm

Available at BSC documentation

2 SCALING TEST

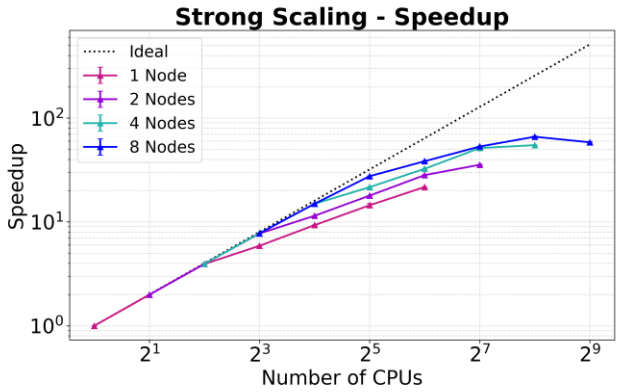
Strong scaling
problem size is fixed and the resources are increased

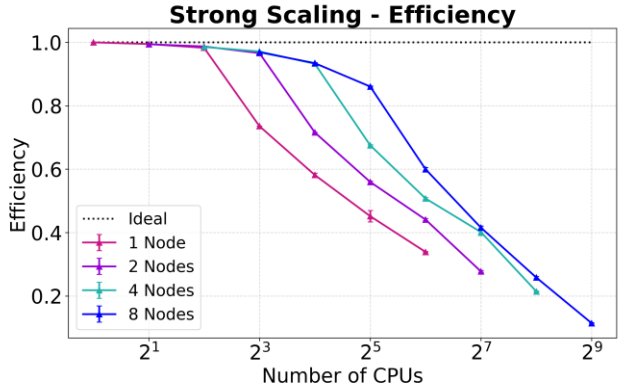
Weak scaling
The problem grows with the number of nodes

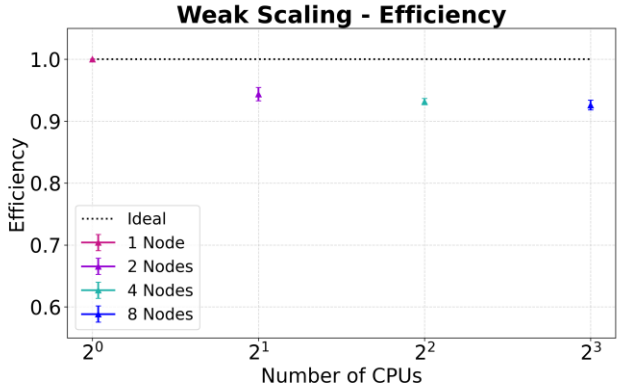
Execution time → Speedup, Efficiency

3 RESULTS

Installation validated







Integration Workflow and CUNQA

An integration workflow was designed

Quantum infrastructure was needed to test our solution → **CUNQA**

1 CUNQA

Infrastructure deployment

The user raises the vQPUs

`qraise`

2 COMPSs

Resource validation

COMPSs resources are requested from SLURM

`enqueue_compss`

`resources.xml`

3 COMPSs

Execution

COMPSs generates an index for each task and the code is executed

4 CUNQA

Resource release

The user releases resources

`qdrop`

Main Modifications

When the **user** submits a job **requesting QPUs**, this **information must reach every layer** of the program. To achieve this, **it was necessary to modify around 100 files.**

1 qpus_per_node

At job submission

New flag to request Quantum Processing Units

`enqueue_compss`

2 QPU processor type

In the code

New type for @constraint: to choose processor type

`@constraint`

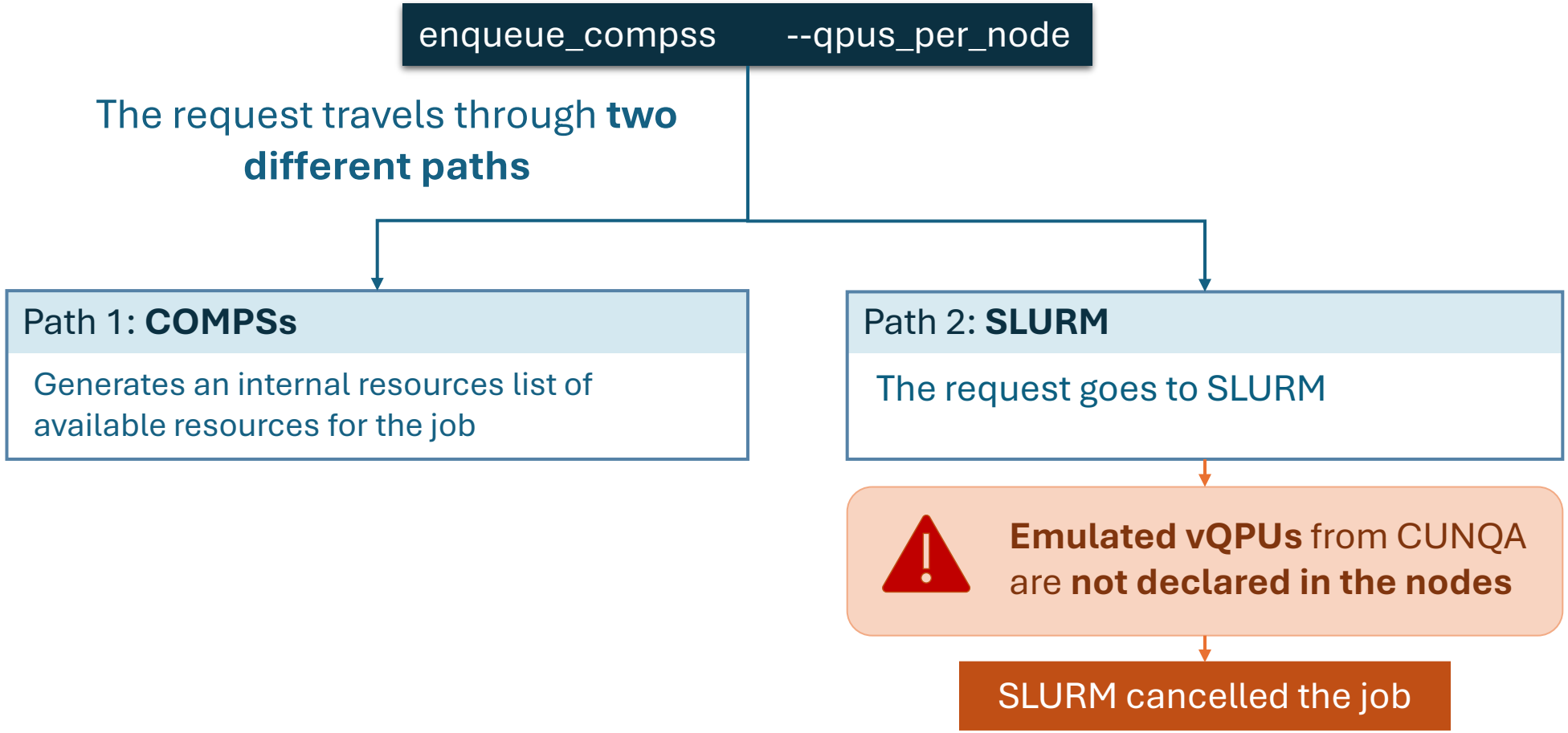
3 QPU Index

At task runtime

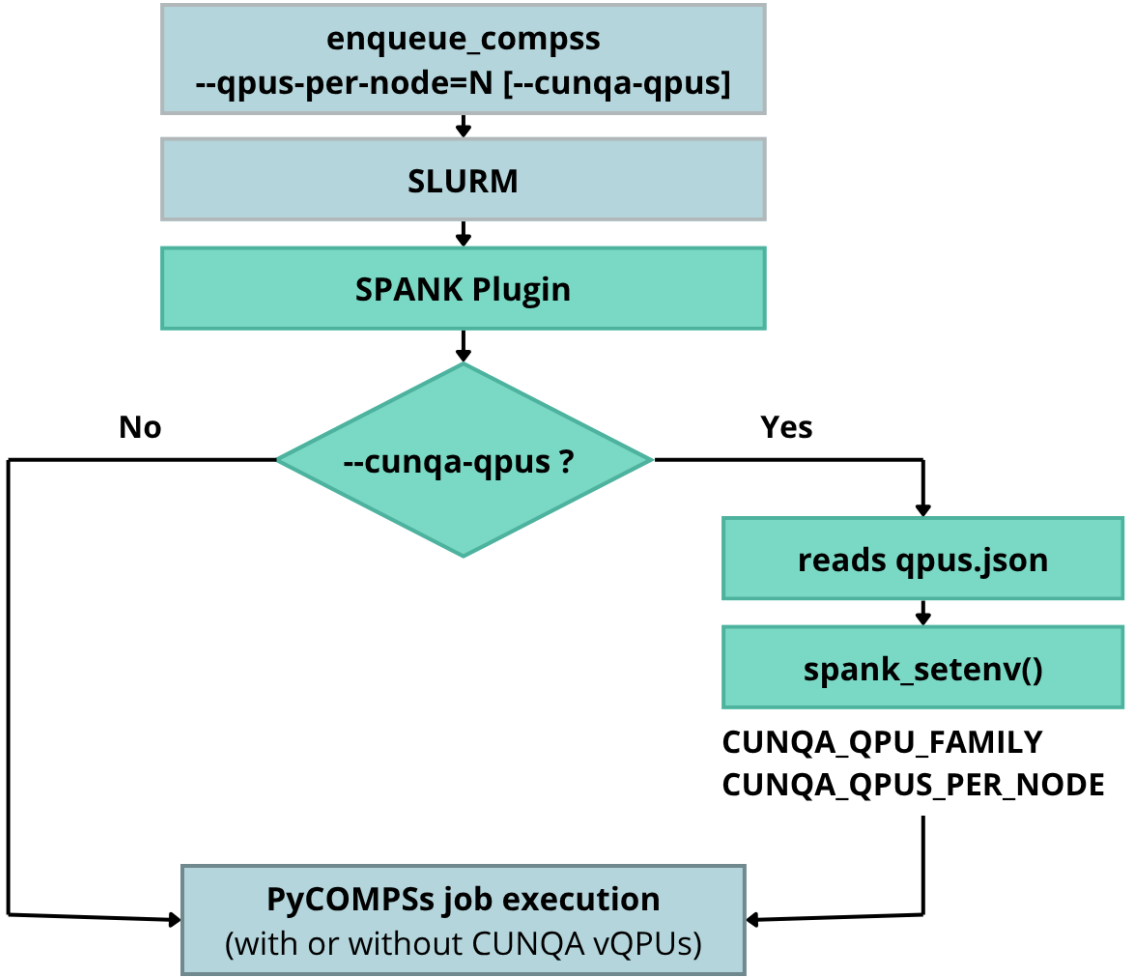
An index for the task to know which vQPU it has been assigned

`COMPSS_BINDED_QPUS`

Making vQPUs Visible to SLURM



SLURM SPANK Plugin

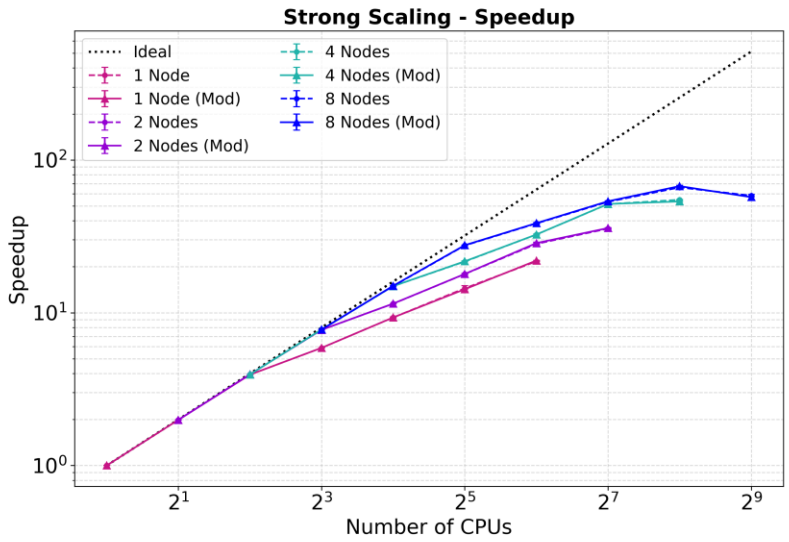


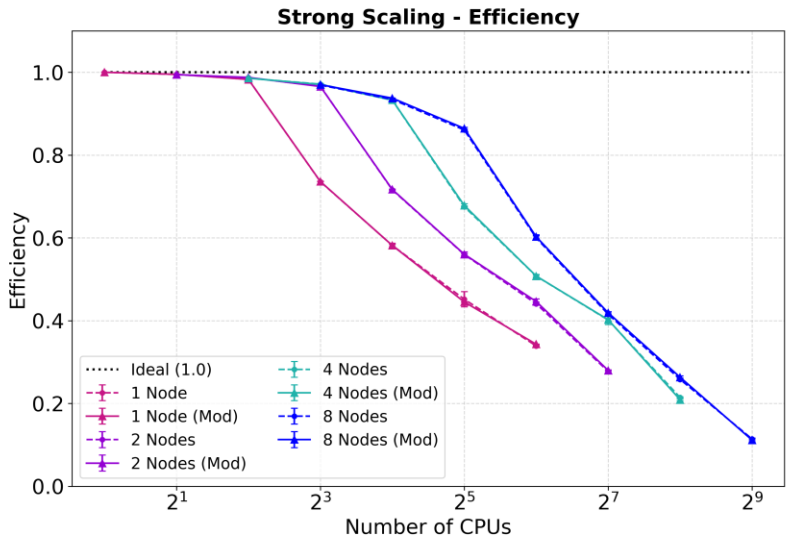
Evaluation (I): No Impact on Classical Users (KMeans)

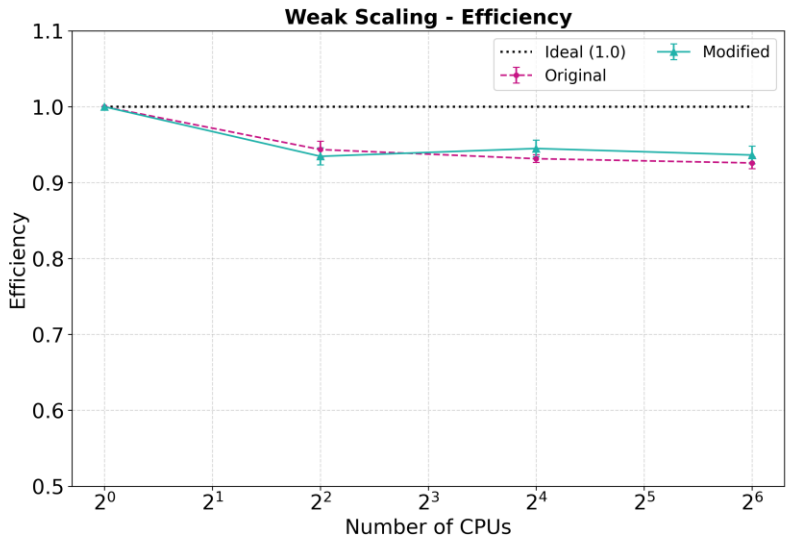
Objective: to make sure that our changes do not affect users who do not use QPUs

Same configurations and metrics were used

No differences were observed between the original and the modified versions





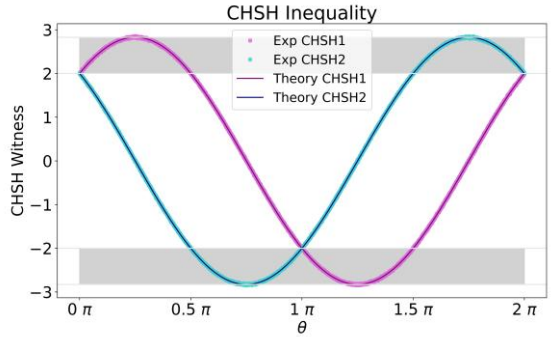


Evaluation (II): Scaling Quantum Tasks on vQPUs (CHSH)

Quantum Benchmark: CHSH inequality

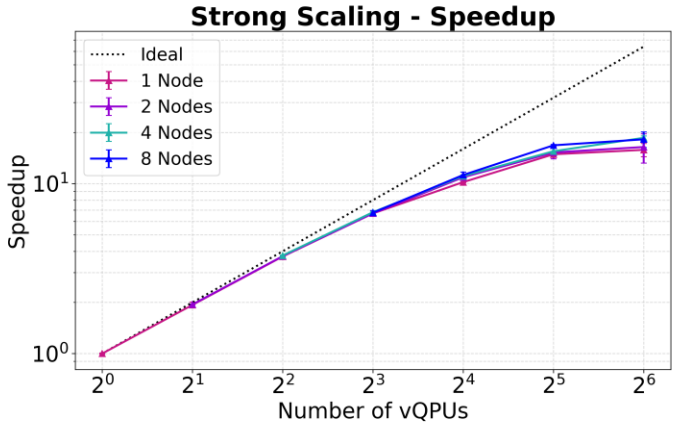
Two entangled qubits, four measurement bases: classical models give at most 2, quantum mechanics up to 2.83

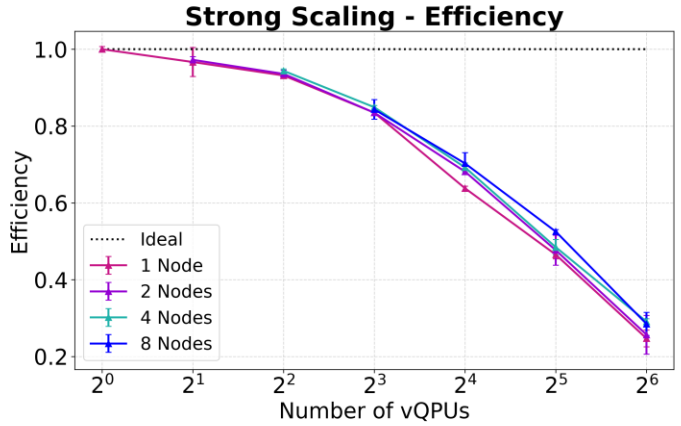
Parallel quantum tasks are scheduled on the vQPUs and scale well

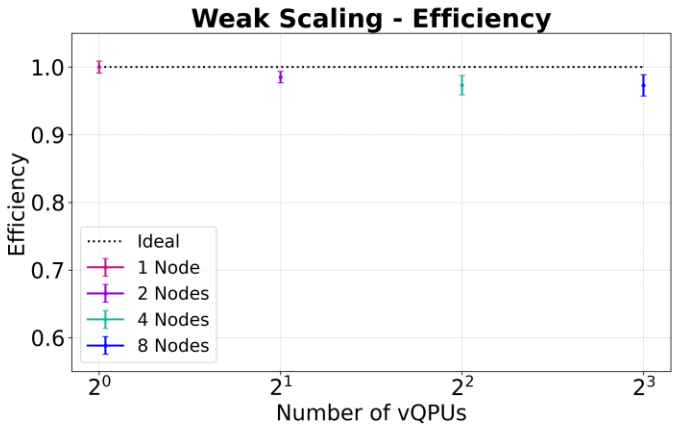


Strong scaling (1024 tasks on 1 to 64 vQPUs)

Weak scaling (128 tasks per vQPU)







Conclusions

- **QPUs in PyCOMPSs**

QPUs as a new processor type. Now PyCOMPSs is a hybrid model.

- **Integrated with CUNQA and SLURM**

SLURM SPANK plugin allows submitting quantum tasks with PyCOMPSs on Qmio.

- **Validated on Qmio**

There is no impact on classical users (KMeans) and satisfactory scalability of parallel quantum tasks (CHSH).

Thank you!



"This work has been financially supported by the Ministry of Economic Affairs and Digital Transformation of the Spanish Government through the QUANTUM ENIA project call - Quantum Spain project, and by the European Union through the Recovery, Transformation and Resilience Plan - NextGenerationEU within the framework of the Digital Spain 2026 Agenda".