Pasqal Defining the quantum reality

Pasqal in HPC centers:
Lessons from Qaptiva and QRMI Deployments, and the Road to Workload Optimization

Oct 15th, Aqadoc 3rd Workshop



Aurélien Nober
VP Software Platform

Agenda

1. About Pasqal
2. Scheduling hybrid jobs in HPC centers
3. How to access our QPUs

This document contains information classified Public and should be handled accordingly

Pasqal Defining the quantum reality

About Pasqal



This document contains information classified Public and should be handled accordingly

Getting out of the labs

Pasqal

2019



INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech

Scientific device debut

2021



PASQAL

R&D device running

2023-2024



Pasqal

- First industrial device on the Cloud
- CEA Genci (France), JSC (Germany) deliveries

TRANSPORTABLE & MODULAR DESIGN

New mechanical design for stable structure

Dedicated electronics

Compact optical setups to ensure stability

Powerful, low noise, stable lasers

Custom software to automatize the QPUs

Quantum Computing Deployed

Pasqal

ALREADY DEPLOYED WITHIN HIGH-PERFORMANCE COMPUTING (HPC) DATA CENTERS





RECENT DELIVERIES

aramco

CINECA

DistriQ
QUANTUM INNOVATION ZONE

ACCESSIBLE VIA THE CLOUD

Microsoft Azure

Google Cloud

Pasqal Cloud

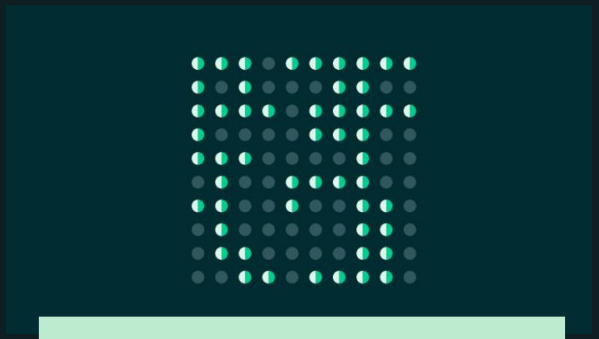
OVHcloud

Scaleway

How does a neutral atom quantum processor work?

Encoding and processing quantum bits, or qubits

Pasqal

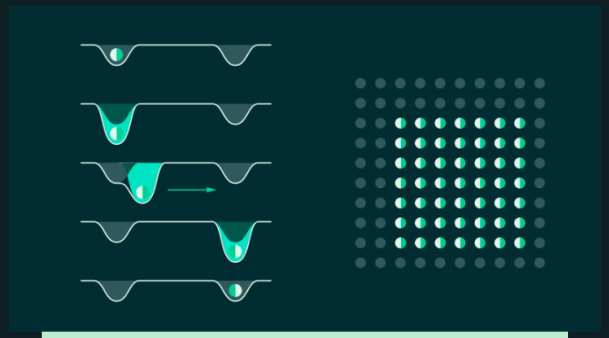


Atom loading

A laser acts like an optical tweezer, trapping atoms

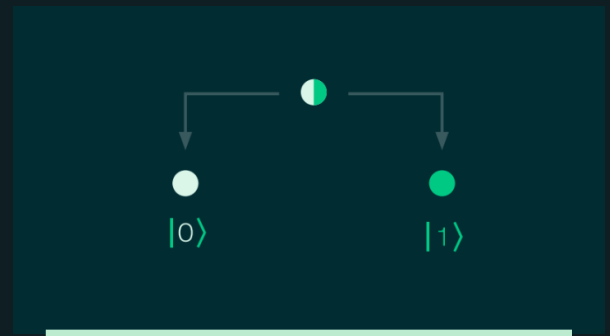
Multiple traps are generated by splitting the laser beam

Half of the traps will contain an atom due to stochastics



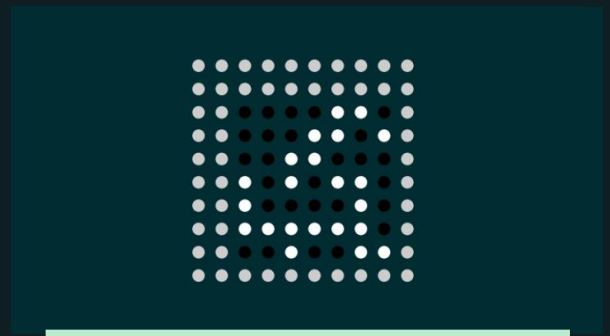
Register preparation

Atoms can be moved to create custom grids, forming the quantum register



Quantum processing

Manipulating the array of atoms with other lasers allows for quantum calculations by switching atoms between energy states



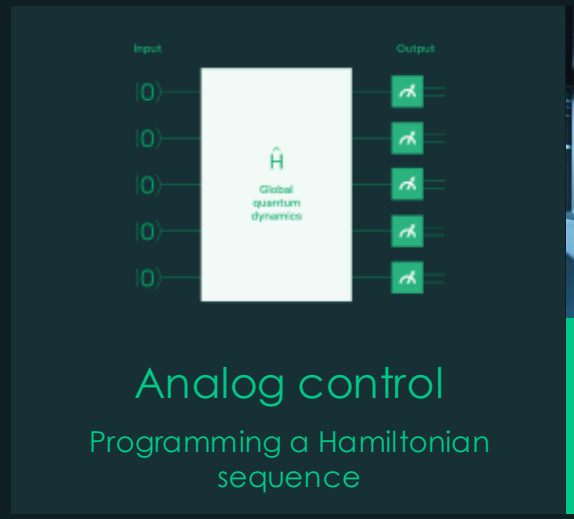
Results read-out

Final results are read by imaging the array: excited atoms (ones) are dark, ground-state atoms (zeros) are bright

Analog and digital simulation

Pasqal

DESIGN OF A NEW CAR



Analog control

Programming a Hamiltonian sequence



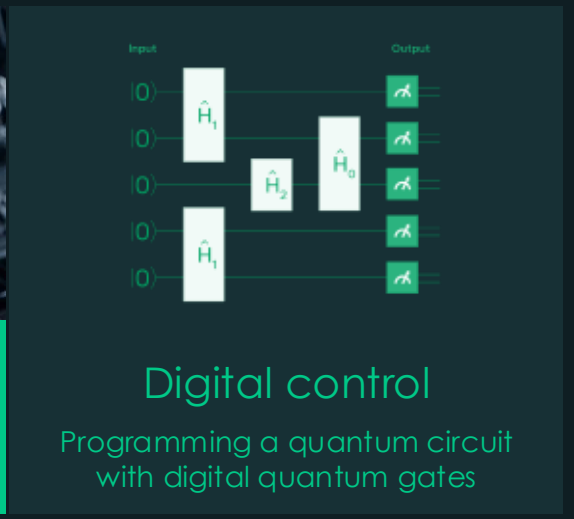
Analog simulation

Wind tunnel



Digital simulation

Computer



Digital control

Programming a quantum circuit with digital quantum gates

CHANGES IN DESIGN

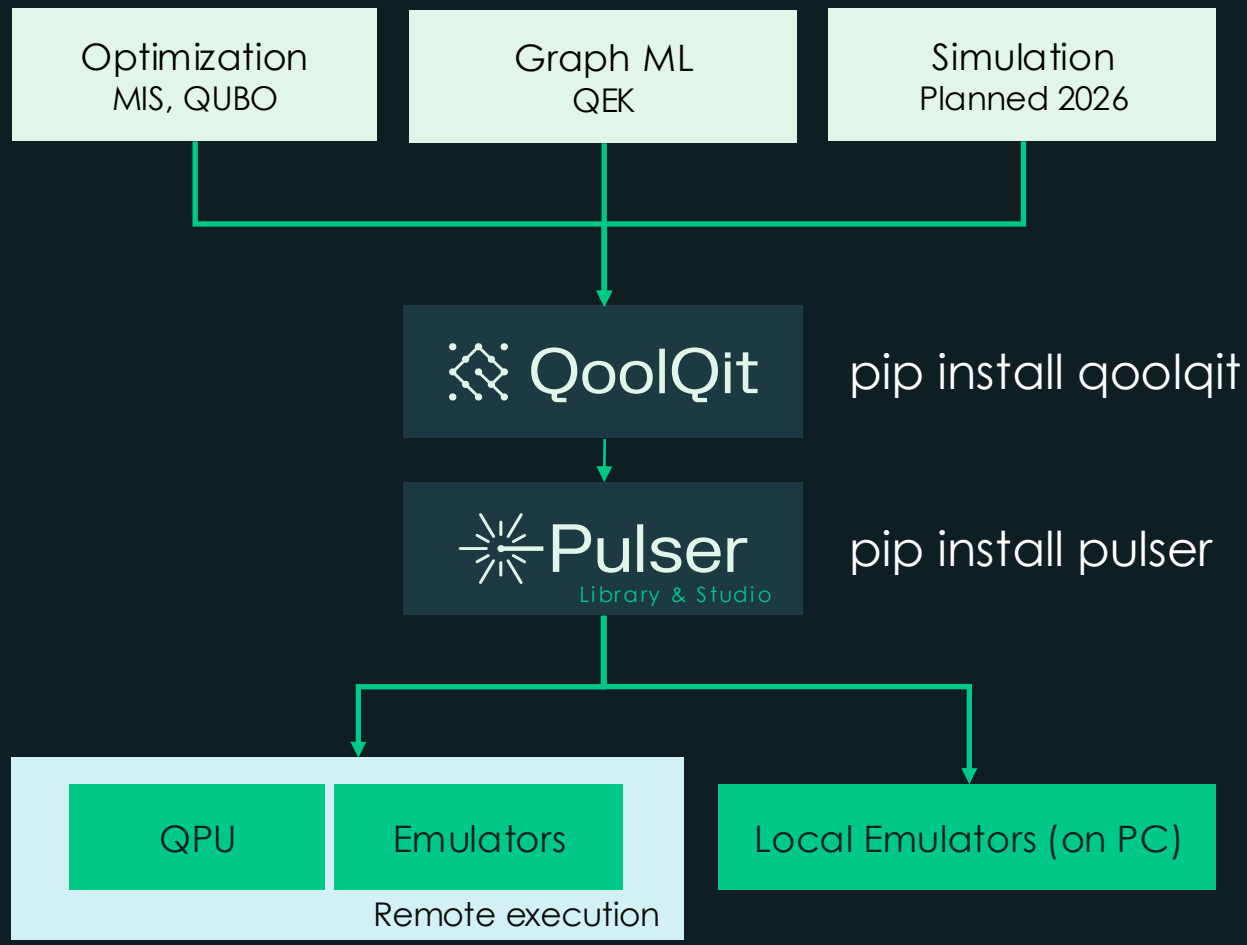
All in-one software suite

Pasqal

Solvers

Quantum SDKs

Execution

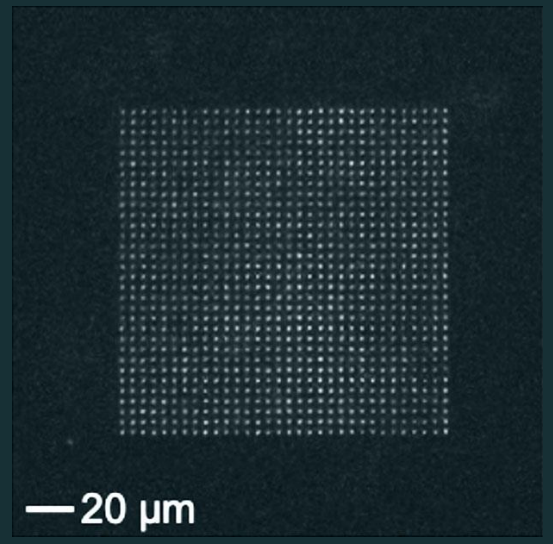


Three major breakthroughs in our roadmap:

1,000 qubits, use case with 2 logical qubits, and quantum advantage in materials

Pasqal

1,000 QUBITS



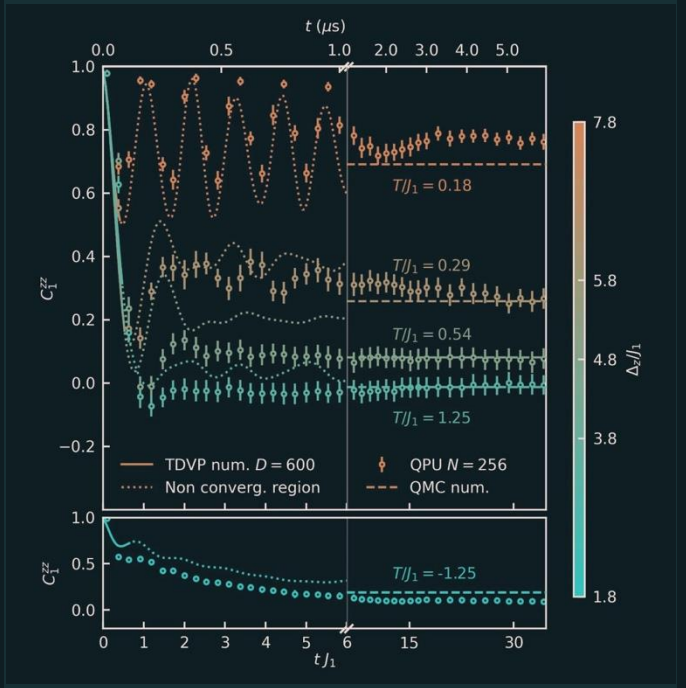
https://arxiv.org/abs/2604.07205

2 LOGICAL QUBITS



https://arxiv.org/abs/2605.21276

QUANTUM ADVANTAGE IN MATERIALS



https://arxiv.org/abs/2603.20372

Our Roadmap

Pasqal

HARDWARE R&D

Physical qubits	200	1,000	10,000		
Base repetition rate	1 Hz	3 Hz	10 Hz	100+ Hz	
Control features	Z-add	Zoning	In-circuit lossless readout	In-circuit reloading	Fast decoders	Large-scale local gates
			10+ PIC enabled parallel 1Q gates		100+ PIC enabled parallel 2Q gates	
					Full addressability	
Logical qubits		2	20		200
Logical fidelity		95%	98%	99.9%	99.9999%
			Break-even ✕		MegaQuOps ✕

HARDWARE PRODUCTS ON-PREM & CLOUD





Pasqal Orion

Alpha	Beta	Gamma
100+ qubits	100+ qubits	140+ qubits



Pasqal Vela
200+ qubits
First QA



Pasqal Centaurus
Analog
& Early FTQC



Pasqal Lyra
Analog
& Impactful FTQC

ALGORITHM & ACCELERATED LIBRARIES

Hybrid Quantum/Classical computing	Slurm integration	Hybrid programming model development			Low-latency HW co-location
Quantum application	QA on Quantum Matter		QA on Optimization & ML		
Open-source libraries	GML apps	Optimization applications	Specialized libraries for Q Simulation, Optimization and Graph ML		Digital FTQC SDK and applications

2022 | 2023 | 2024 | 2025 | 2026E | 2027E | 2028E | 2029E ›

Note: Based on management's reasonable estimates with respect to product research and development.

Pasqal Defining the quantum reality

Scheduling hybrid jobs in HPC centers



This document contains information classified Public and should be handled accordingly

Pasqal

HPC - Qaptiva environment



A small glance at Qaptiva platform

Bull

Pasqal

myQLM
Quantum programming stack

- Python API w/ C++ source code
- Compilation
- Job abstraction
- Emulation backends

Submit jobs

Qaptiva access
Orchestration and execution

- Job management
- Access control
- Backend integration
- Unified interface

Schedule jobs

Qaptiva HPC/appliance
Classical emulation

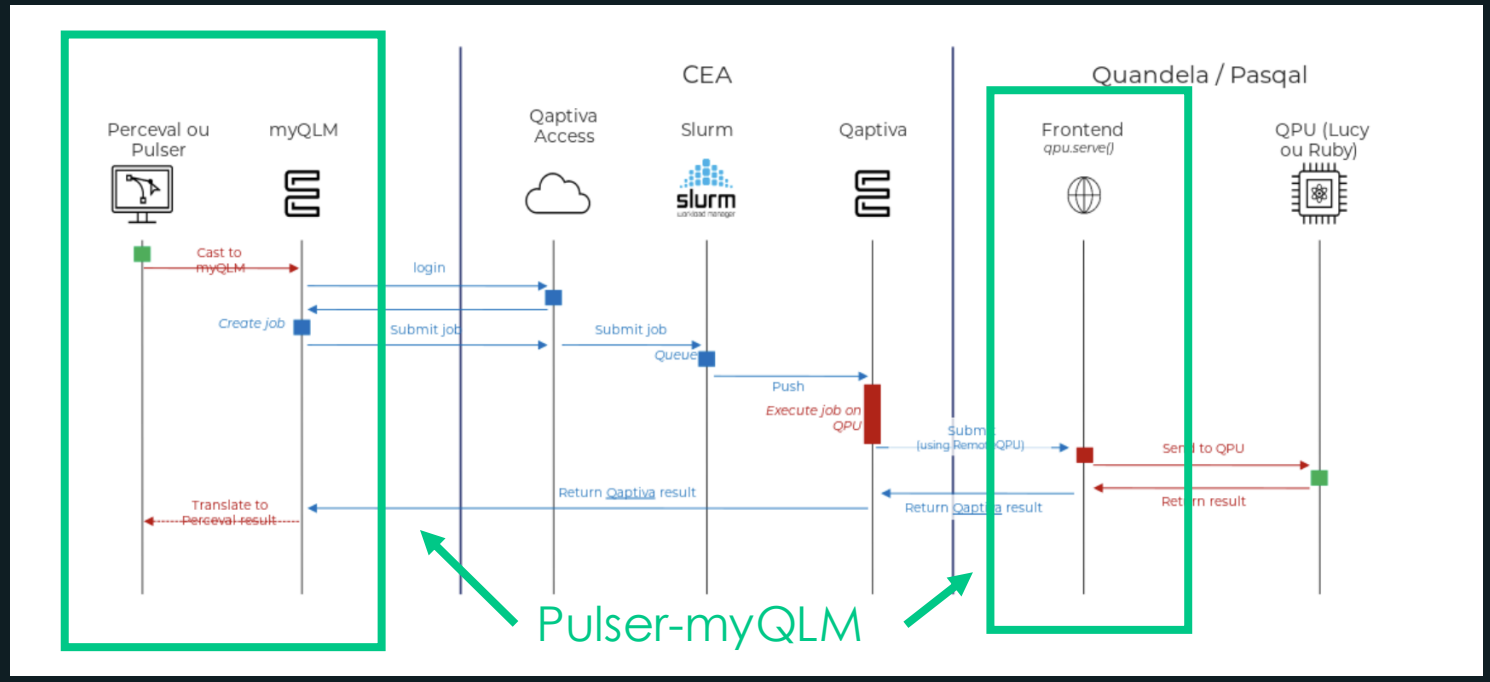
- Dedicated classical resources
- Distributed architecture
- Slurm integration

How Pulser connects to Qaptiva at TGCC (and JSC)

Pasqal

Pulser-MyQLM

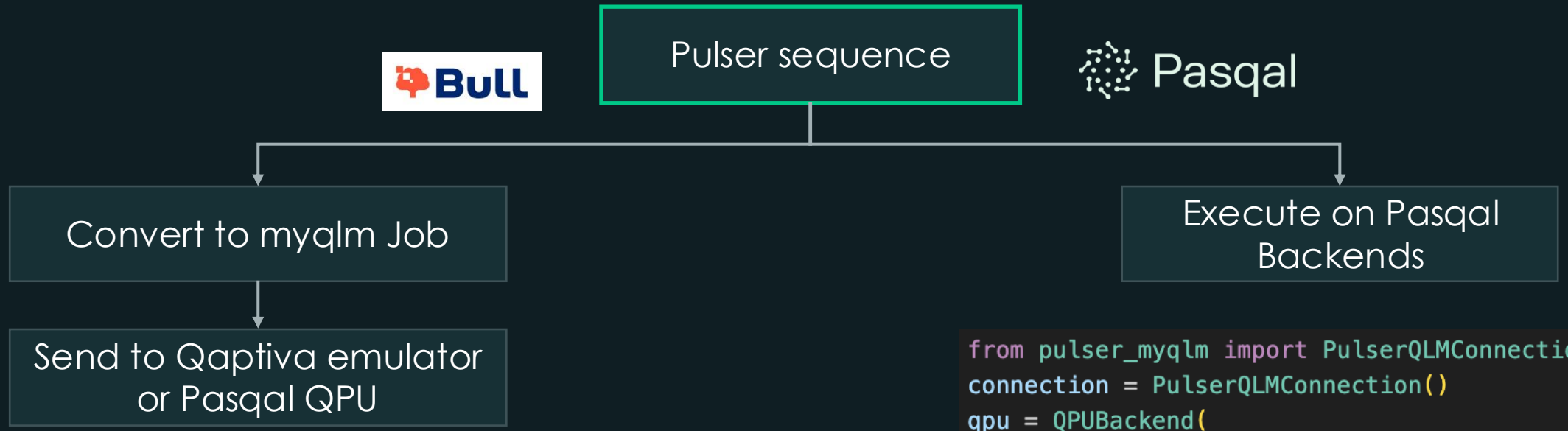
- is an Open-Source python package
- converts Pulser Sequence into myQLM Job for emulation
- Connects the Qaptiva to PasqalQPU
- Connects Pasqal software to the Qaptiva



Source: CEA TGCC, www-hpc.cea.fr/tgcc-public/en/html/toc/QuantumSoftwareStack.html

How to send programs to a Pasqal QPU

Pasqal




```python
from qat.qlmaas import QLMaaSConnection
from pulser_myqlm import IsingAQPU
conn = QLMaaSConnection()
myqlm_job = IsingAQPU.convert_sequence_to_job(
    pulser_sequence,
    nbshots=NBSHOTS,
    modulation=MODULATION
)
async_results = conn.get_qpu("qat.qpus:AnalogQPU")().submit(myqlm_job)
results = async_results.join()
```

```python
from pulser_myqlm import PulserQLMConnection
connection = PulserQLMConnection()
qpu = QPUBackend(
    pulser_sequence,
    connection=connection
)
remote_results = qpu.run(
    [JobParams(runs=1000, variables=[])],
    wait=True      You, 2 minutes ago • Uncommit
)
count = remote_results[0].final_bitstrings
```

HPC - QRMI environment

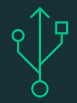


Goals of our HPC-QC software integration

Pasqal



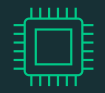
Hybrid workloads enablement



Seamless QPU integration



Open-source software



Hardware-agnostic when possible



Unchanged user experience

Toward a user-centric HPC-QC



Resources optimization



Resources management, no compilation

Separation of concerns

A glance at our concrete integration at CINECA

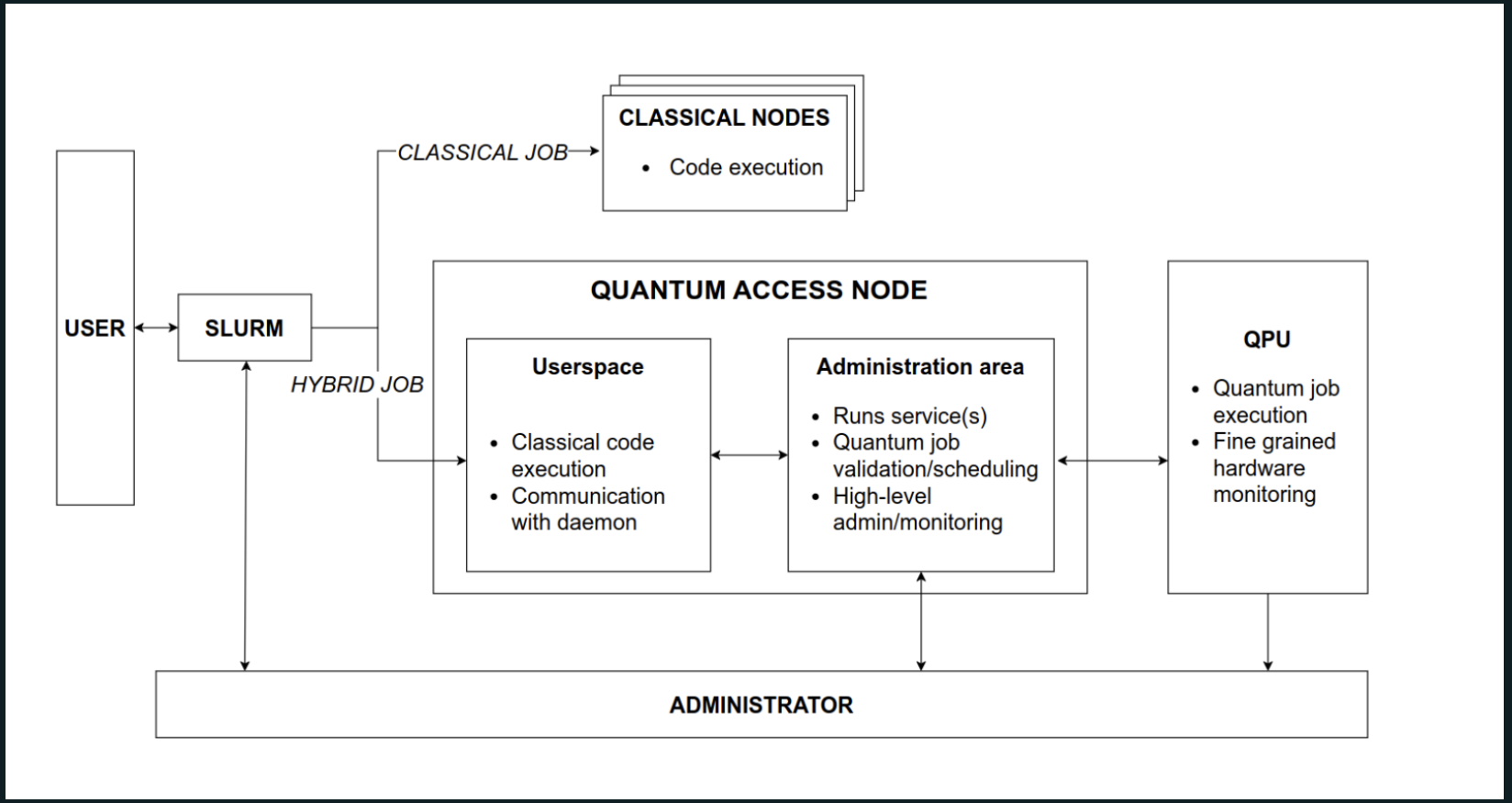
Pasqal

A dedicated compute resource

- Quantum Access Node (QAN)

Three extra software components

- Slurm SPANK plugin
- Quantum Resource Management Interface (QRMI) library
- Warden, Pasqal middleware

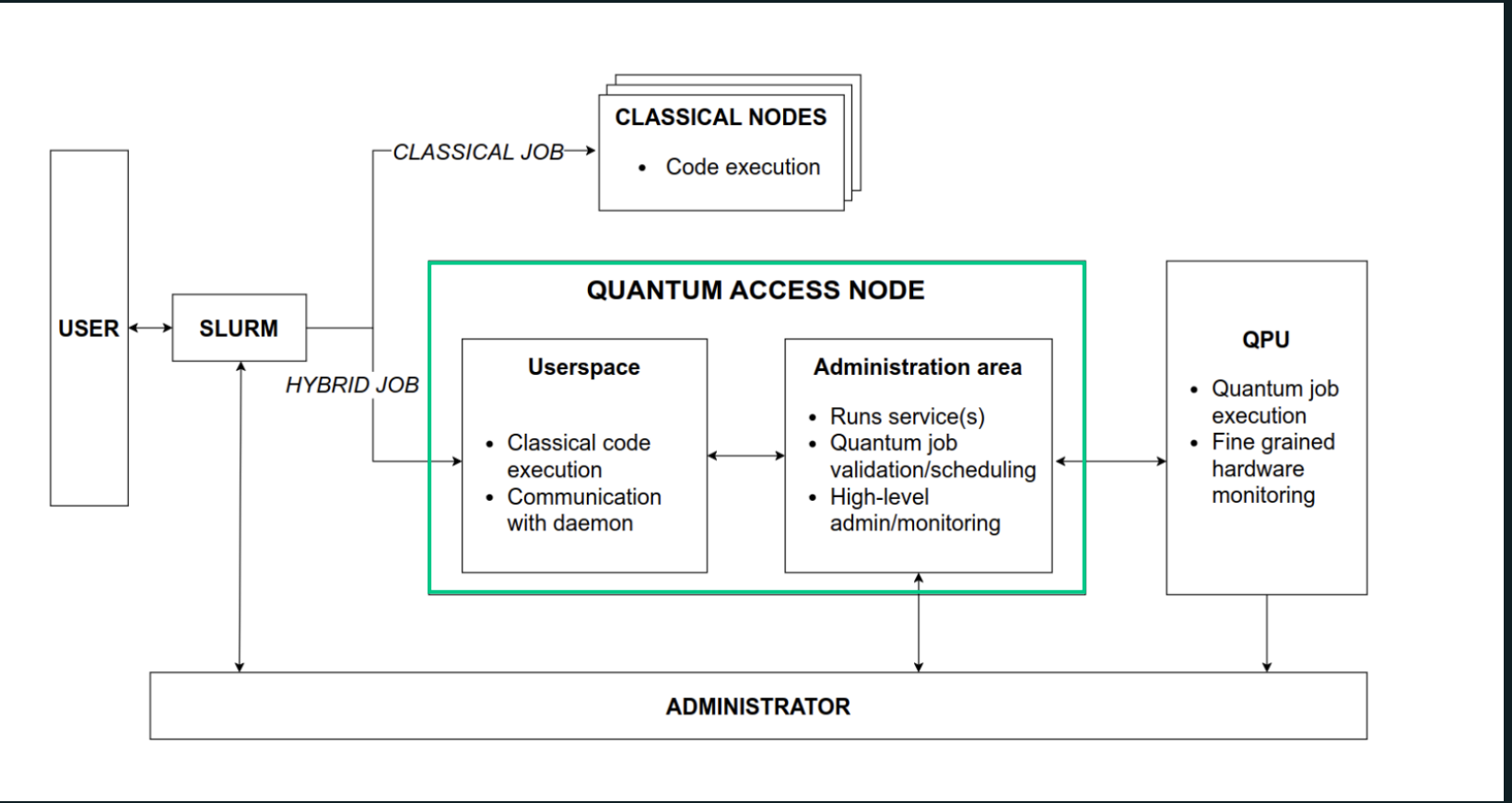


Source: A. Wennersteen et al., "Towards a user-centric HPC-QC environment", SC Workshops '25, arXiv:2509.20525

The Quantum Access Node (QAN) compute resource

Pasqal

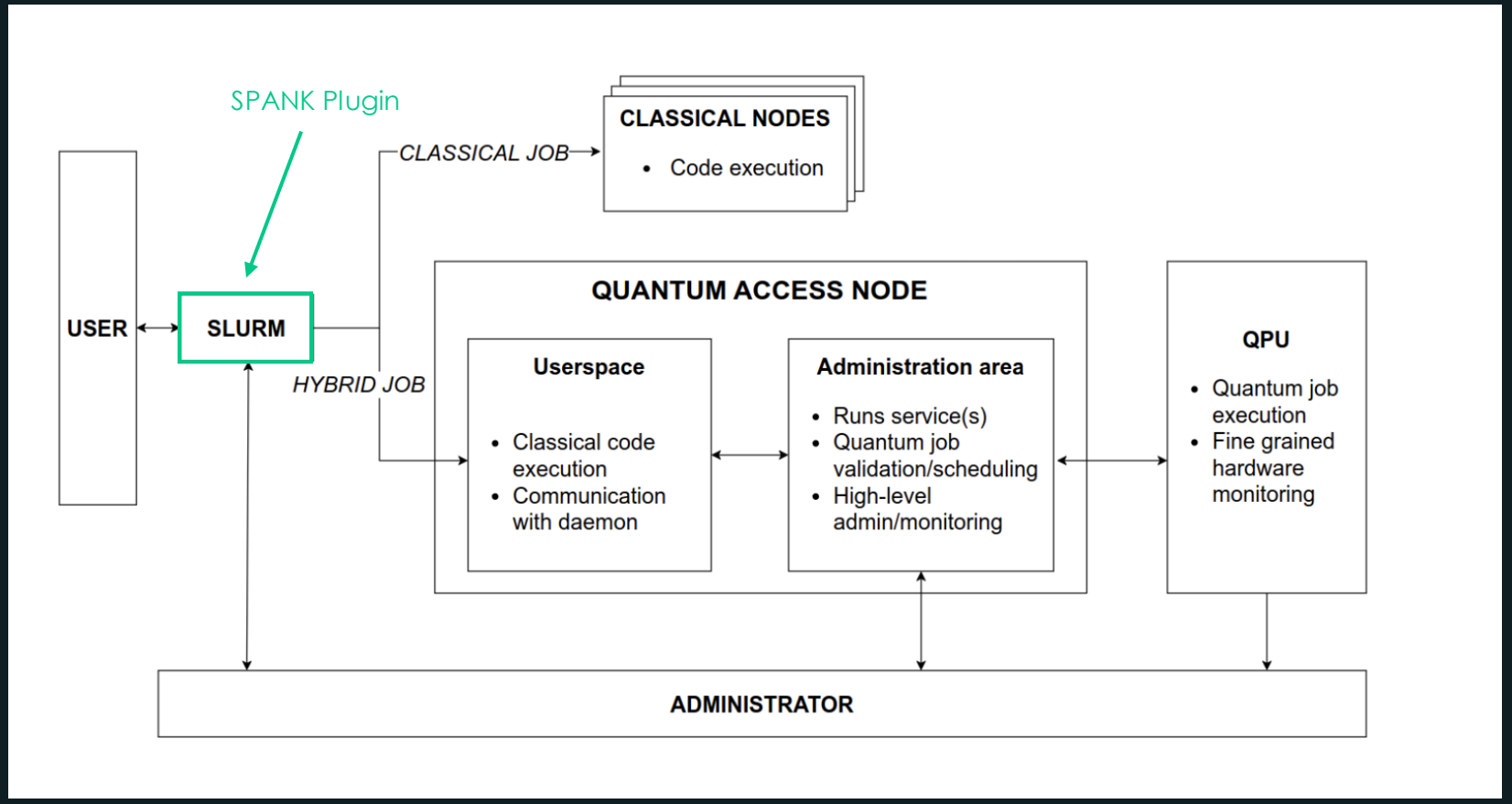
- **Any node** which can be reserved by the user to access the QPU is a Quantum Access Node
- A QAN also includes **classical resources for hybrid jobs**
- The design is **agnostic of the network integration between the QAN and the QPU.** It can accommodate low bandwidth/high latency for neutral-atoms or stronger needs for other technologies (e.g. superconducting)
- The QAN can rely on **hardware provided by the HPC center or the quantum hardware vendor**



Source: A. Wennersteen et al., "Towards a user-centric HPC-QC environment", SC Workshops '25, arXiv:2509.20525

Slurm Plug-in Architecture for Node and job Kontrol (SPANK)

Pasqal



Source: A. Wennersteen et al., "Towards a user-centric HPC-QC environment", SC Workshops '25, arXiv:2509.20525

Plugin enables quantum resource reservation workflow

Pasqal

Admin defines QPU resources in *qrmi_conf.json*

↓

User selects the QPU with --qpu=QPU

↓

Slurm checks QPU availability and acquires the resource

↓

Slurm injects connection env variables in job environment at prolog

↓

Slurm releases the quantum resources after job completion

qiskit-community / **spank-plugins**

```
{
"resources": [
    {
    "name": "FRESNEL",
    "type": "pasqal-cloud",
    "environment": {
        "QRMI_PASQAL_CLOUD_PROJECT_ID": "<PROJECT_ID>"
        "QRMI_PASQAL_CLOUD_AUTH_TOKEN": "<AUTH_TOKEN>"
        }
    },
    {
    "name": "PASQAL_LOCAL",
    "type": "pasqal-local",
    "environment": {
        "QRMI_URL": "http://localhost:8006"
        }
    }
    ]
}
```

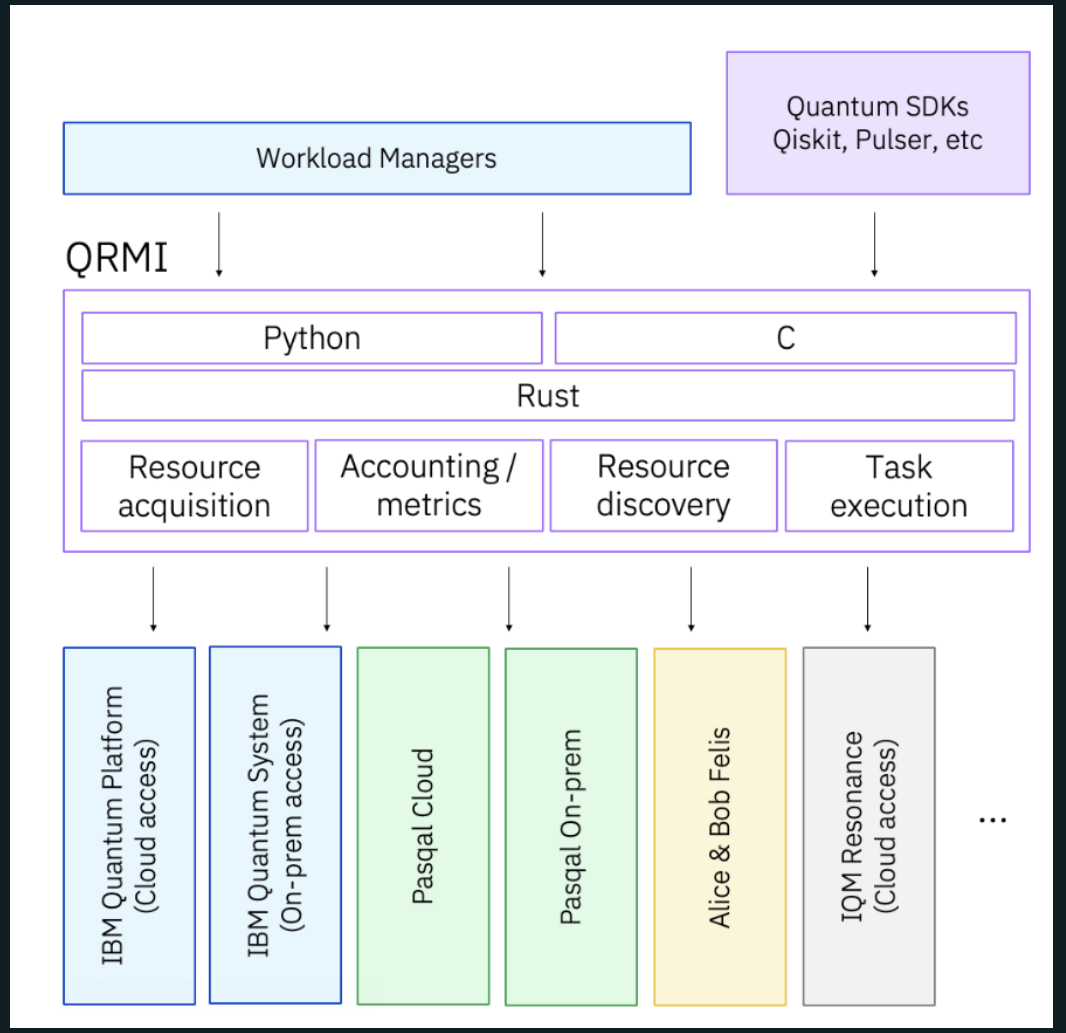
qrmi_conf.json

A few words about the QRMI architecture

Pasqal

The **Quantum Resource Management Interface (QRMI)** is a thin, vendor-agnostic middleware layer that lets HPC centers treat quantum computers as first-class, schedulable resources.

- **Unified abstract resources interface** across vendors (e.g. is_accessible, submit_job etc…)
- Handles **on-prem** setup as well as **cloud** bursting
- Source code is Rust based with bindings in C, Python and Lua
- Integrates with **multiple workload managers** such as Slurm, Flux, Grid Engine, k8s etc…
- Native **programming sdk integration** (e.g. qiskit, pulser)

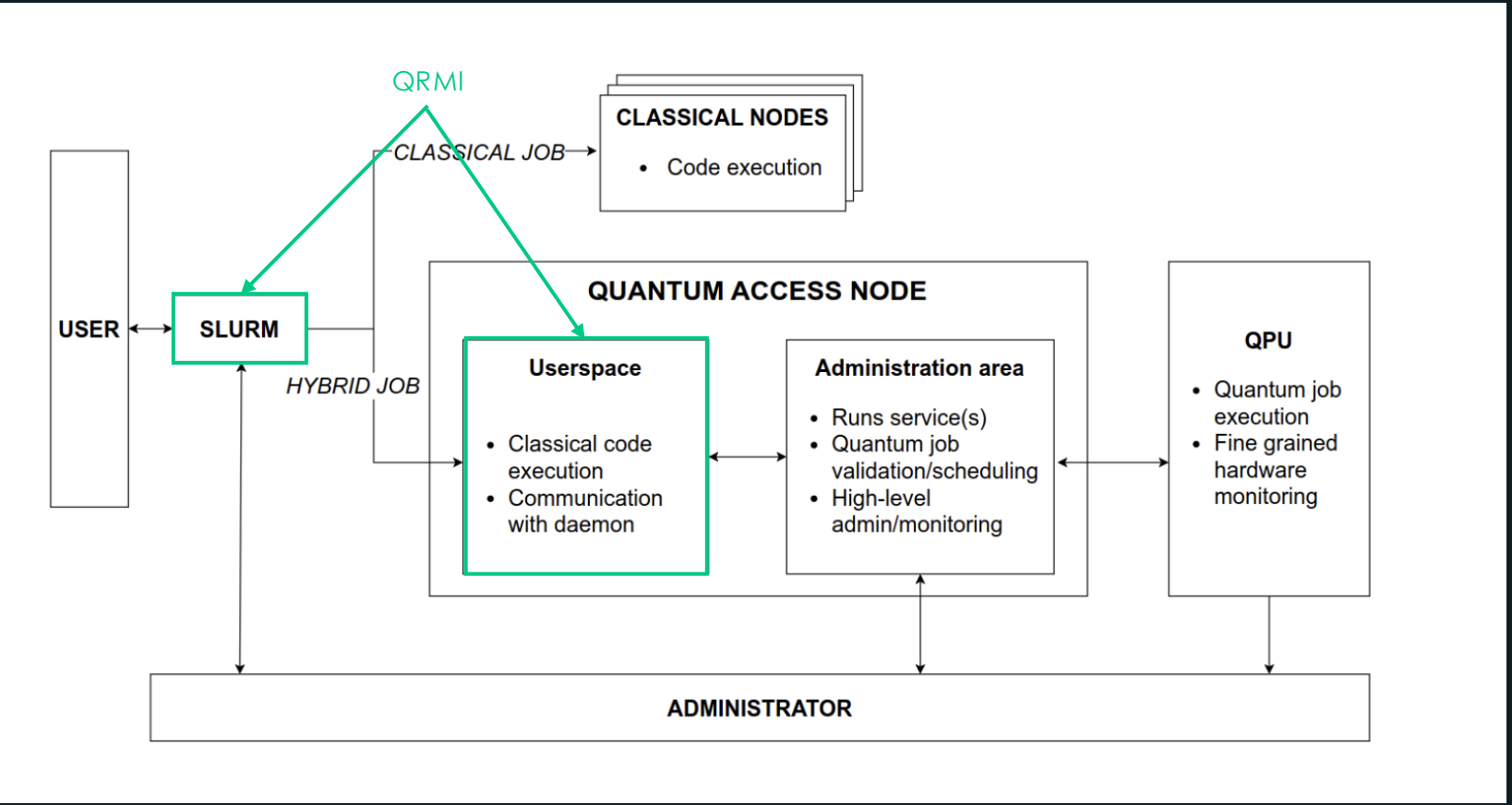


Source: T. Badts et al., "Examining QRMI as a Unified Interface for Quantum-HPC Integration", arXiv:2607.19591, 2026

QRMI connects scheduler and application to quantum resources

Pasqal

- QRMI is used in the **Slurm SPANK Plugin** to acquire/release the quantum resource
- QRMI is used in the **client software stack** to standardize interactions between the user and quantum resources
  - PulserQRMIConnection
  - NVIDIA Cuda-Q integration
  - IBM Qiskit Pasqal provider

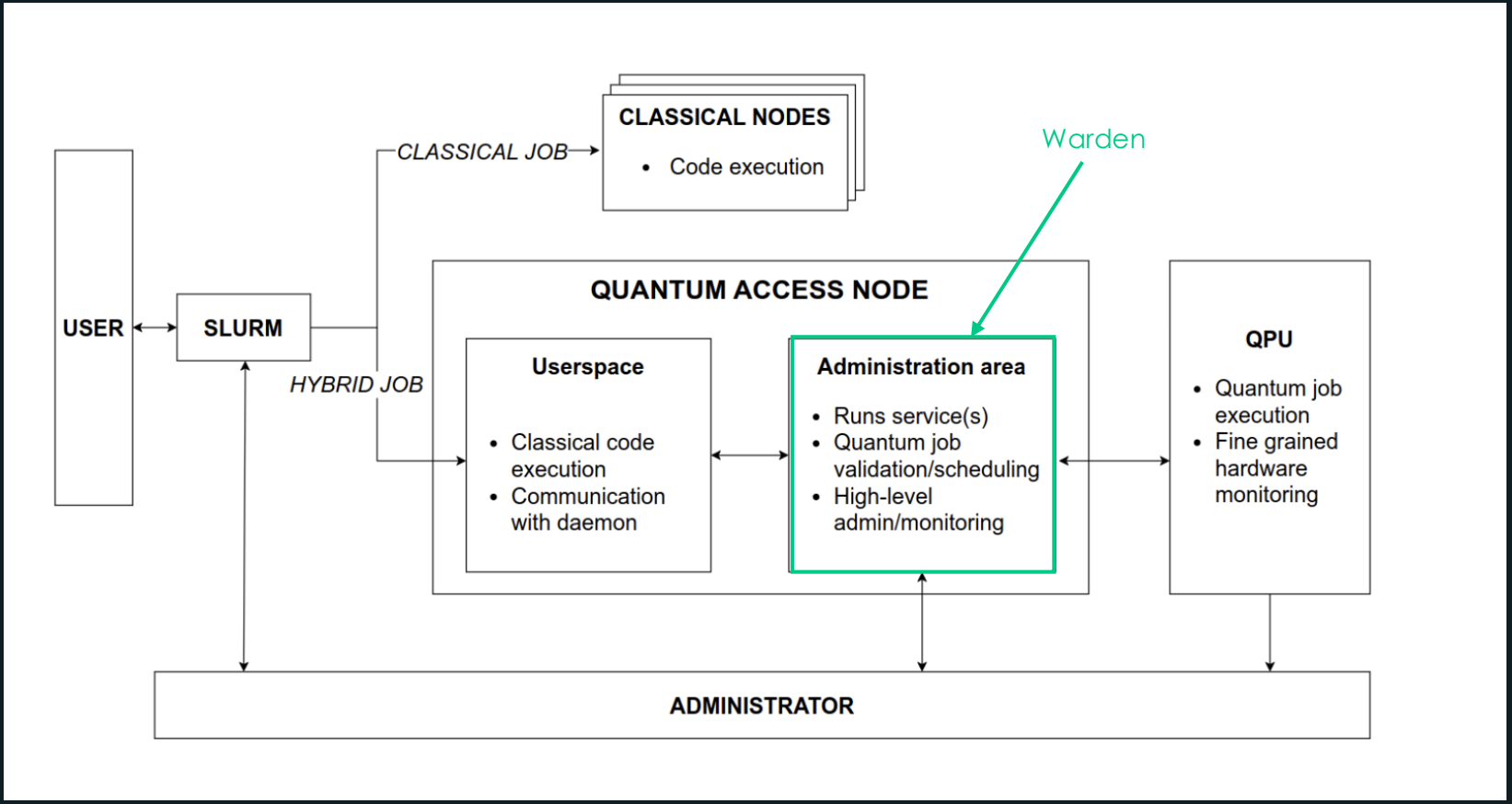


Source: A. Wennersteen et al., "Towards a user-centric HPC-QC environment", SC Workshops '25, arXiv:2509.20525

Warden: QAN<>Pasqal QPU middleware

Pasqal

- Features:
  - Authentication
  - Configurable scheduling policy
  - Secure QPU administration
  - Accounting

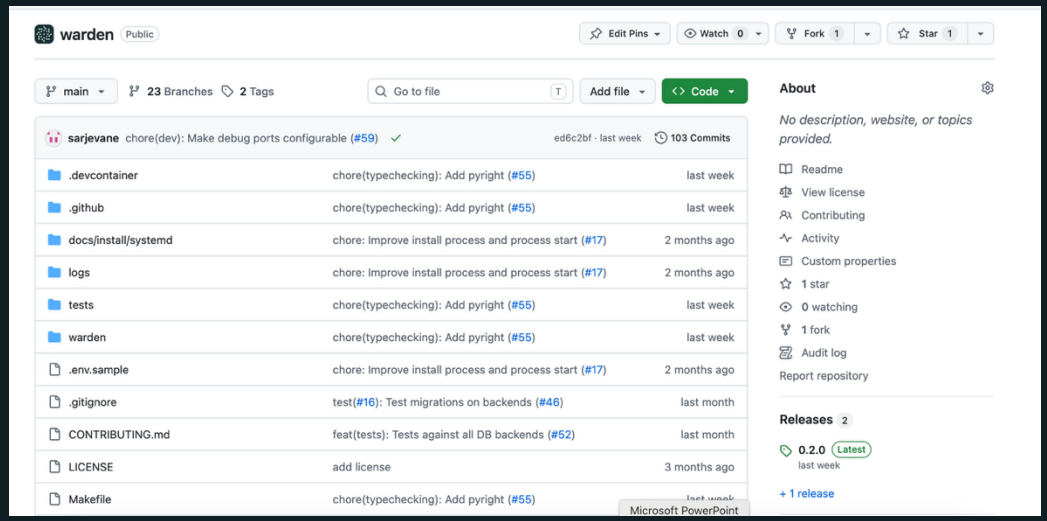


Source: A. Wennersteen et al., "Towards a user-centric HPC-QC environment", SC Workshops '25, arXiv:2509.20525

Warden is nimble

Pasqal

- **Requirements:** python3.11+
- **Configuration** made with a simple yaml file



pasqal-io/warden

Authentication

Warden supports **munge** for authentication since it's Slurm default solution. Other protocols can be integrated

Database

Warden supports various databases (inc. **MariaDB**) for quantum jobs and user sessions

Scheduling

A customizable scheduling policy with FIFO and Weighted FIFO as default for quantum jobs.

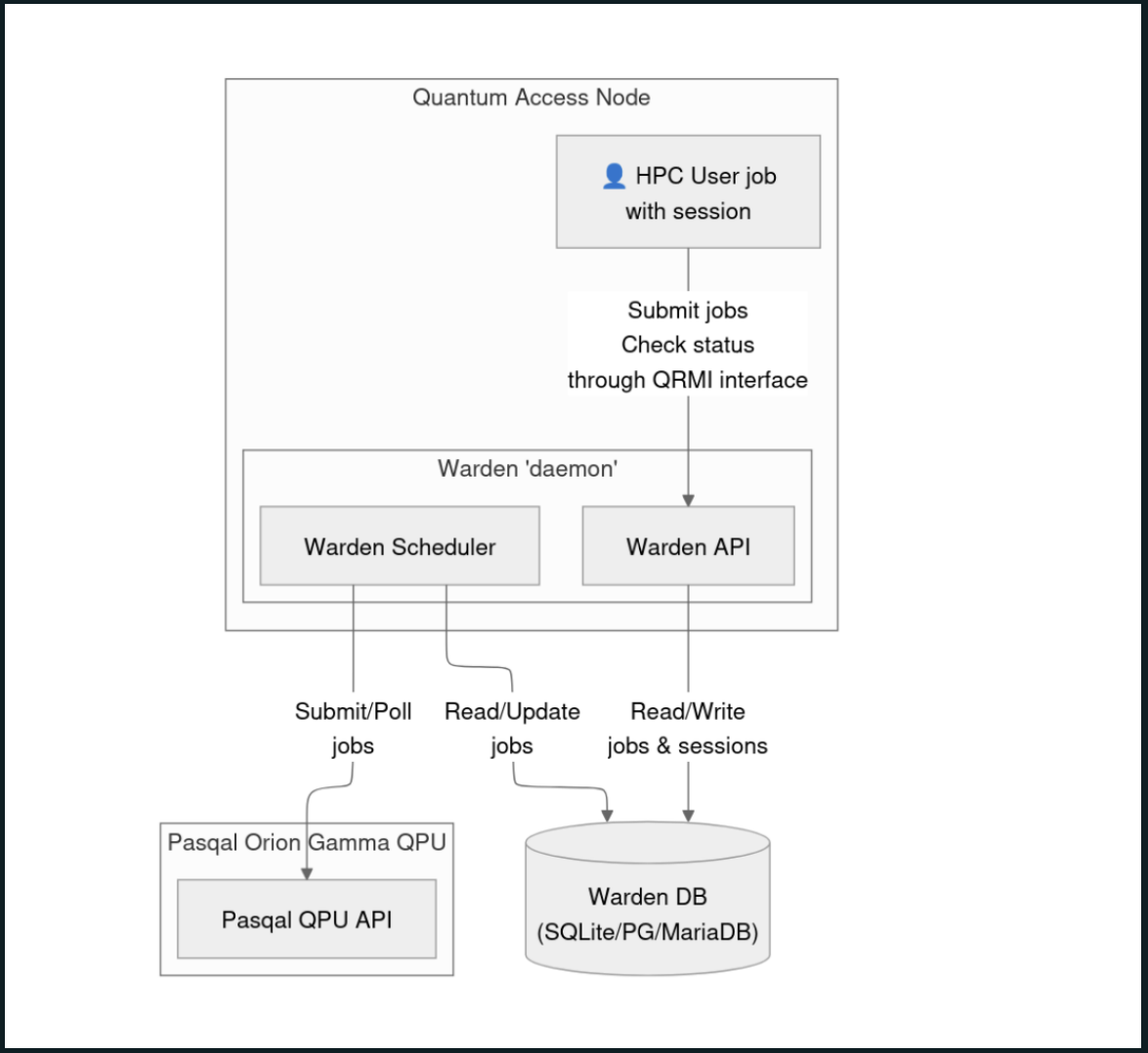
API

Warden exposes job scheduling capabilities inc. time tracking, qpu description and status

Warden architecture

Pasqal

- **API:** receives requests from the QRMI, handles sessions, stores jobs in a database
- **Scheduler:** queues jobs from the database onto the Pasqal QPU



Source: Pasqal, github.com/pasqal-io/slurm-qrmi-warden-hpc-training

Further challenges 1/2

Pasqal

- SLURM assumes relatively stable and schedulable resources, usually statically attached to a node, while near term QPUs (across vendors) are:
  - Interrupted by calibration jobs
  - Favoring technological breakthrough over stability between generation
- SPANK plugins are **not officially supported** by SchedMD which can be a blocker for some HPC centers

	GRES	HRES	Dynamic licenses
Resource model	Per-node resource declared in gres.conf	License-like counts on node groups, defined in resources.yaml	Cluster-wide counter, stored in slurmdbd
Changing capacity	Edit config, then reconfigure or restart daemons	Config file; runtime updates only in recent releases	Live update with sacctmgr, pushed to slurmctld
Downtime	Drain or down the whole host node	Disable a layer or lower its count	Set count to 0 or reserve the license
Race conditions	Low: node state lives in slurmctld; draining does not stop running jobs	Runtime changes lost on restart or reconfigure	Count lags QPU state: jobs can start before QPU goes down/calibrating
Node locality	Job lands on the node hosting the QPU	Job limited to nodes in the QPU's layer	None: needs a separate node constraint
Maturity	Stable, widely used for GPUs	Beta since Slurm 25.05	Stable since Slurm 23.02

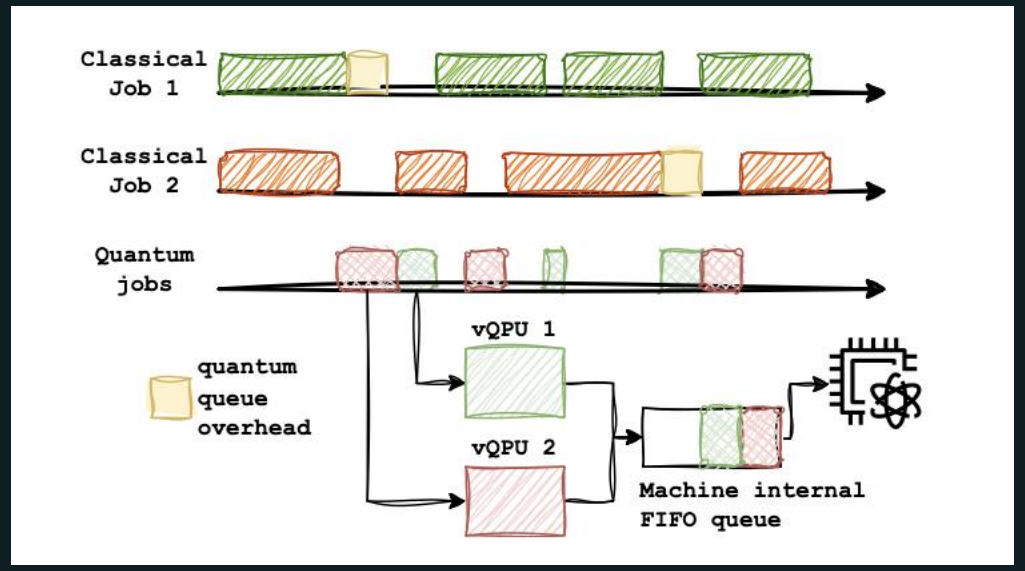
Source: T. Badts et al., "Examining QRMI as a Unified Interface for Quantum-HPC Integration", arXiv:2607.19591, 2026

Further challenges 2/2

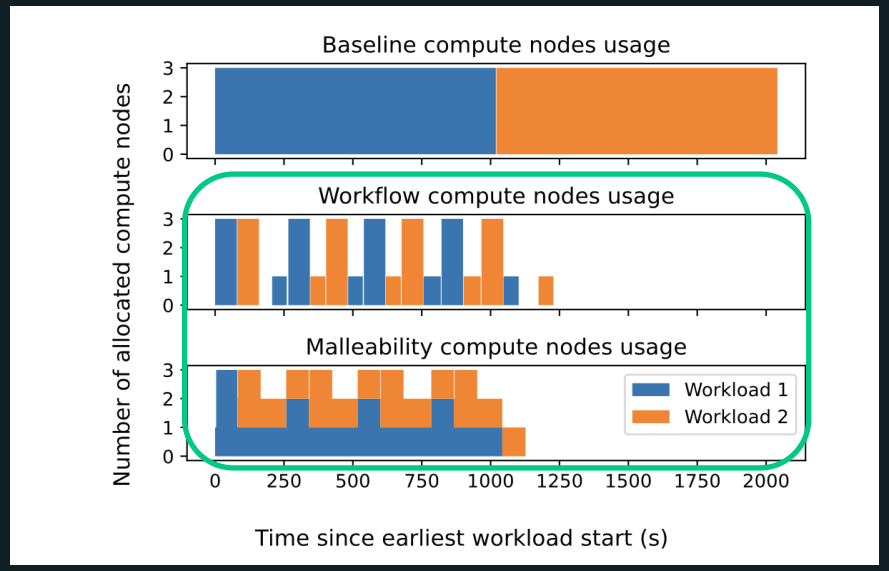
Pasqal

- The existence of an internal QPU queue hidden from the workload manager's scheduling system (the two-queue problem) creates sub-optimal job orchestration and waste of classical resources"
- Malleability and workflow managers can release idle classical nodes during quantum phases: **much lower classical resource usage** for a **slight increase in wall time**
- This experiment was done on heterogeneous but fixed patterns of hybrid QC

Classical nodes sit idle during quantum waits



Malleability and workflows free idle nodes



Source: M. Cipollini et al., "Three ways to share a QPU: Scheduling strategies for hybrid Quantum-HPC applications", arXiv:2604.14955, 2026

Pasqal

Workload manager-QRMI integration survey

Workload managers in HPC centers

Pasqal

Workload managers examined

Current maintainer and year of first release

Scheduler	Maintained by	First released
Slurm	SchedMD NVIDIA-owned since 2025	2001 LLNL and partners
PBS	Altair part of Siemens	1991 NASA Ames
LSF	IBM acquired Platform, 2012	1992 Platform Computing
Grid Engine	HPC Gridware Open Cluster Scheduler	1993 as CODINE
Kubernetes *Orchestrator*	CNCF Linux Foundation	2014 Google
Flux *Graph scheduler*	LLNL US DOE national lab	2014 LLNL

QRMI comparison framework

Three design patterns used to rate each workload manager

1 **Treat quantum resources as schedulable**

Define the QPU as a custom resource and track its live availability, so jobs request it like a GPU.

2 **Thin integration layer for lifecycle**

Hooks or plugins map job events to QRMI: acquire at setup, execute, release at completion.

3 **Encapsulate complexity in config and middleware**

Provider details and credentials live in config files and QRMI, so the scheduler interface stays simple.

Source: T. Badts et al., "Examining QRMI as a Unified Interface for Quantum-HPC Integration", arXiv:2607.19591, 2026

Traditional Batch schedulers – QRMI compatibility

Pasqal

What all four share

Slurm, PBS, LSF and Grid Engine

Same acquire → execute → release pattern

Hooks acquire the QPU at job setup, pass its details to the job and release it at job end.

Dynamic availability, exposed to race conditions

Each polls QPU state (dynamic licenses, server_dyn_res, ELIM, load sensors), so counts can lag the real QPU.

Where PBS stands out

Availability check before allocation

PBS checks the QPU before reserving nodes

The runjob hook runs on the PBS server before other resources are allocated. If the QPU is unavailable, the job goes back to the queue and no nodes sit idle.

Slurm, LSF and Grid Engine acquire after allocation, so nodes can idle while the QPU is busy.

Where they differ

Selected integration details from Table III

	Slurm	PBS	LSF	Grid Engine
Extension language	C shared library	Python script	Any language	Any (Go preferred)
Code update	Rebuild per node	Re-register in qmgr	Redeploy scripts	Rebuild hooks
Job rejection	[Limited]: plain error code	[Flexible]: rejection w/ custom attributes	Hook error codes	JSV: reject or modify

Source: T. Badts et al., "Examining QRMI as a Unified Interface for Quantum-HPC Integration", arXiv:2607.19591, 2026, Sections IV–V and Table III

Flux – QRMI compatibility

Pasqal

1 Hierarchical resource model

Fluxion models the cluster as a graph, with QPUs at rack level, one per vendor. A job is a resource shape matched by graph traversal, filtered by vendor or device.

2 Compatible with other workload managers

Fluxion runs standalone through a Go SDK. Fluence plugs it into Kubernetes as a scheduler plugin; an analogous design is planned for Flux in HPC.

3 Fluence tackles one instance of the two-queue problem

One producer submits the quantum task while N-1 workers stay gated until it nears the top of the queue, cutting worker idle time.

- **Relies on the quantum resource exposing its status and queue position**

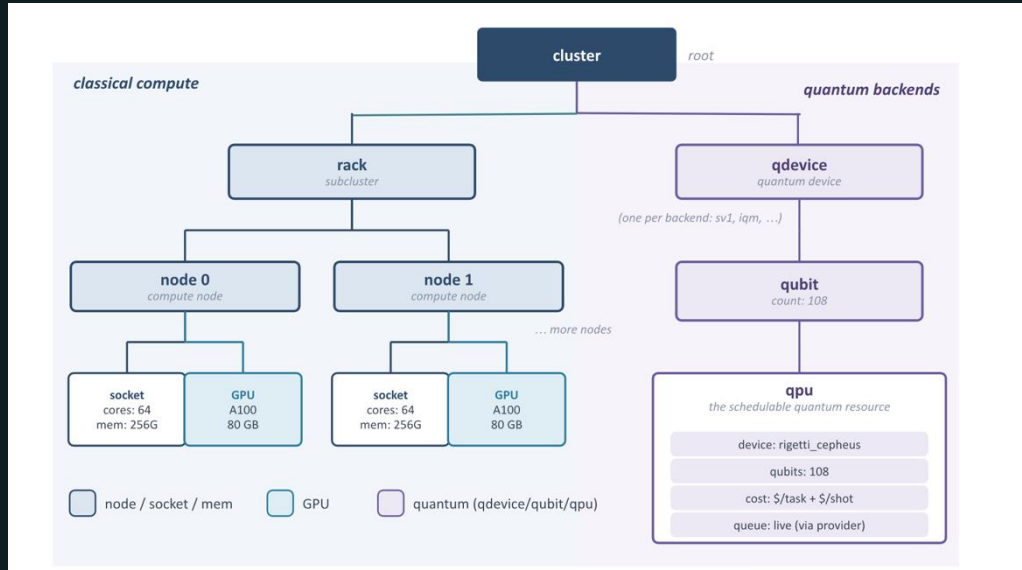


Fig. 2. **Hybrid modeling of classical and quantum resources** in the Fluxion resource graph. A quantum device is modeled at the same level as a rack, and job requests for work correspond to a traversal in the graph. Constraints are used to limit search to a resource class.

Source: V. Sochat and D. Milroy, "Hybrid Quantum and Classical Workload Management with Graph-based Scheduling", arXiv:2607.09151, 2026, Sections 1–2 and Fig. 2

Future directions

Pasqal

01 **QRMI interface extension for**

- Queue information and configuration
- QPU processing time accounting

02 **Extend FIFO queue to Weighted FIFO queue for fair share queuing**

03 **Uniformization of Cloud and On-prem QPUs interface**

04 **Monitoring**

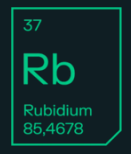
05 **Compilation toolchain**

Pasqal Defining the quantum reality

How to access our QPUs

This document contains information classified Public and should be handled accordingly

Access our QPUs via Pasqal Cloud



Access 100+ Qubit QPUs

Run hybrid quantum-classical workloads on Pasqal Orion, our 100+ qubit neutral-atom QPU

Test our suite of Emulators

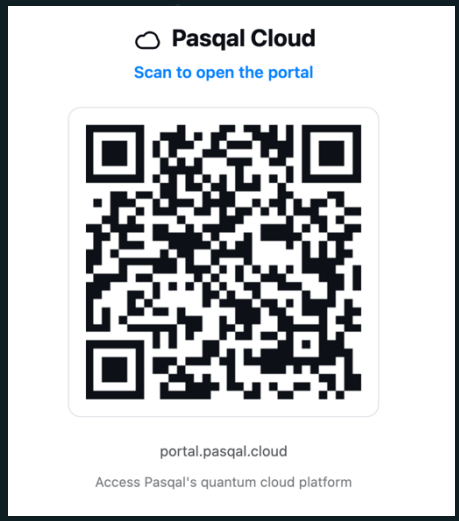


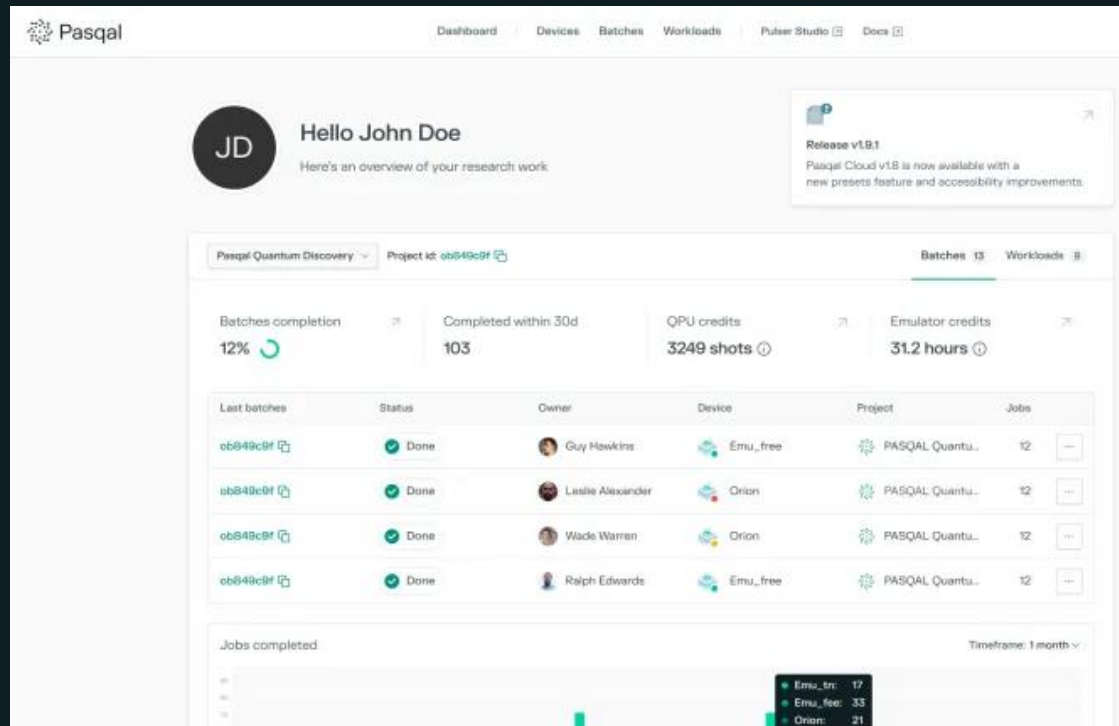
Test and refine your quantum algorithms using our high-performance emulators before running them on real QPUs

Open-Source Software Stack and SDKs



Pasqal's comprehensive open-source tools and developer SDKs to build and test quantum applications seamlessly on local machines or in the cloud





This document contains information classified Public and should be handled accordingly

Open Access to Ruby at TGCC

Pasqal's 100 qubits neutral atom quantum computer

WHO CAN APPLY?

EU based researchers in academia and industry

WHAT CAN YOU APPLY FOR?

Up to 100 free hours of quantum computing time

ANY REQUIREMENTS?

One, only open research is allowed

Find out how to access Quantum Computing resources.

https://www.pasqal.com/resources/open-access-to-pasqal-quantum-computer-at-tgcc/



©Copyright CEA / Cadam 2024

Pasqal

Thank you!

This document contains information classified Public and should be handled accordingly