Workload and workflow management in quantum-classical HPC

Yoonho Park
Principal Research Scientist
IBM Quantum

Munetaka Ohtani
Senior Software Engineer
IBM Quantum

Shweta Salaria
Research Scientist
IBM Quantum

https://github.com/ohtanim/SCA-HPCAsia-2026

Tutorial Agenda

Time		Theme	Presenter
1:30pm - 1:35pm	5min	Opening	Yoonho Park
1:35pm - 2:45pm	70min	Towards the Integration of HPC and Quantum Computing	Munetaka Ohtani
2:45pm - 3:15pm	30min	Coffee Break	
3:15pm - 4:25pm	70min	Workflow and Workload Management in Quantum-Classical HPC	Shweta Salaria
4:25pm - 4:30pm	5min	Closing	Yoonho Park

Towards the Integration of HPC and Quantum Computing

Munetaka Ohtani
Senior Software Engineer
IBM Quantum

Tutorial Agenda

Time		Theme	Presenter
1:30pm - 1:35pm	5min	Opening	Yoonho Park
1:35pm - 2:45pm	70min	Towards the Integration of HPC and Quantum Computing	Munetaka Ohtani
2:45pm - 3:15pm	30min	Coffee Break	
3:15pm - 4:25pm	70min	Workflow and Workload Management in Quantum-Classical HPC	Shweta Salaria
4:25pm - 4:30pm	5min	Closing	Yoonho Park

Goals

- Understand the need and motivation behind Quantum-HPC Integration.

- Understand how to use these heterogeneous technologies together, building on already widely adopted tools and solutions.

Background

Growing Momentum in Quantum-HPC and Quantum-Classical Integration

"Quantum Processing Units (QPUs) are also beginning to integrate into hybrid infrastructures."
"Its findings emphasize … the exponential growth of research in this domain.

The evolution of HPC: how AI and quantum computing are reshaping supercomputing
https://link.springer.com/article/10.1007/s11227-025-07241-7

Achieving a practical quantum advantage for near-term applications is widely expected to rely on hybrid classical-quantum algorithms.

arXiv "Hybrid Classical-Quantum Supercomputing: A demonstration of a multi-user, multi-QPU and multi-GPU environment"

The convergence of high-performance computing (HPC) and quantum computing (QC) is emerging as a pivotal strategy for tackling problems that demand computational capabilities beyond the reach of classical systems alone.

Defining quantum-ready primitives for hybrid HPC-QC supercomputing: a case study in Hamiltonian simulation
https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2025.1528985/full

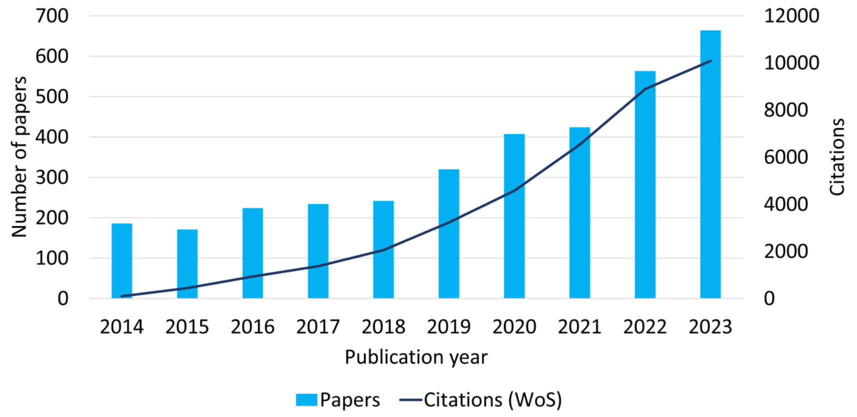


FIGURE 2. Distribution of publications and citations on QC and HPC by year (2014-2023).

Mapping the Landscape of Quantum Computing and High Performance Computing Research Over the Last Decade
https://ruja.ujaen.es/server/api/core/bitstreams/57396cf4-3ee0-4505-89b3-33b5ee3610f1/content

Quantum Resource Management
Key Challenges

Integration & Scheduling Gaps

- No native QPU support in workload managers such as Slurm
- Hybrid workflows are difficult to coordinate across distributed systems
- Multiple schedulers increase operational complexity and overhead
- Lack of standardized APIs across quantum providers complicates integration

Usability & Abstraction Issues

- Inconsistent Resource Models for cloud/on-premise QPUs
- Manual scripts are fragile and difficult to scale
- Limited usability for HPC users unfamiliar with quantum computing
- Difficulty benchmarking hybrid HPC-QPU workload

Visibility & Security Limitations

- No unified monitoring or accounting for quantum jobs
- Limited transparency in QPU cost/performance metrics
- Fragmented security models and access controls for cloud QPUs

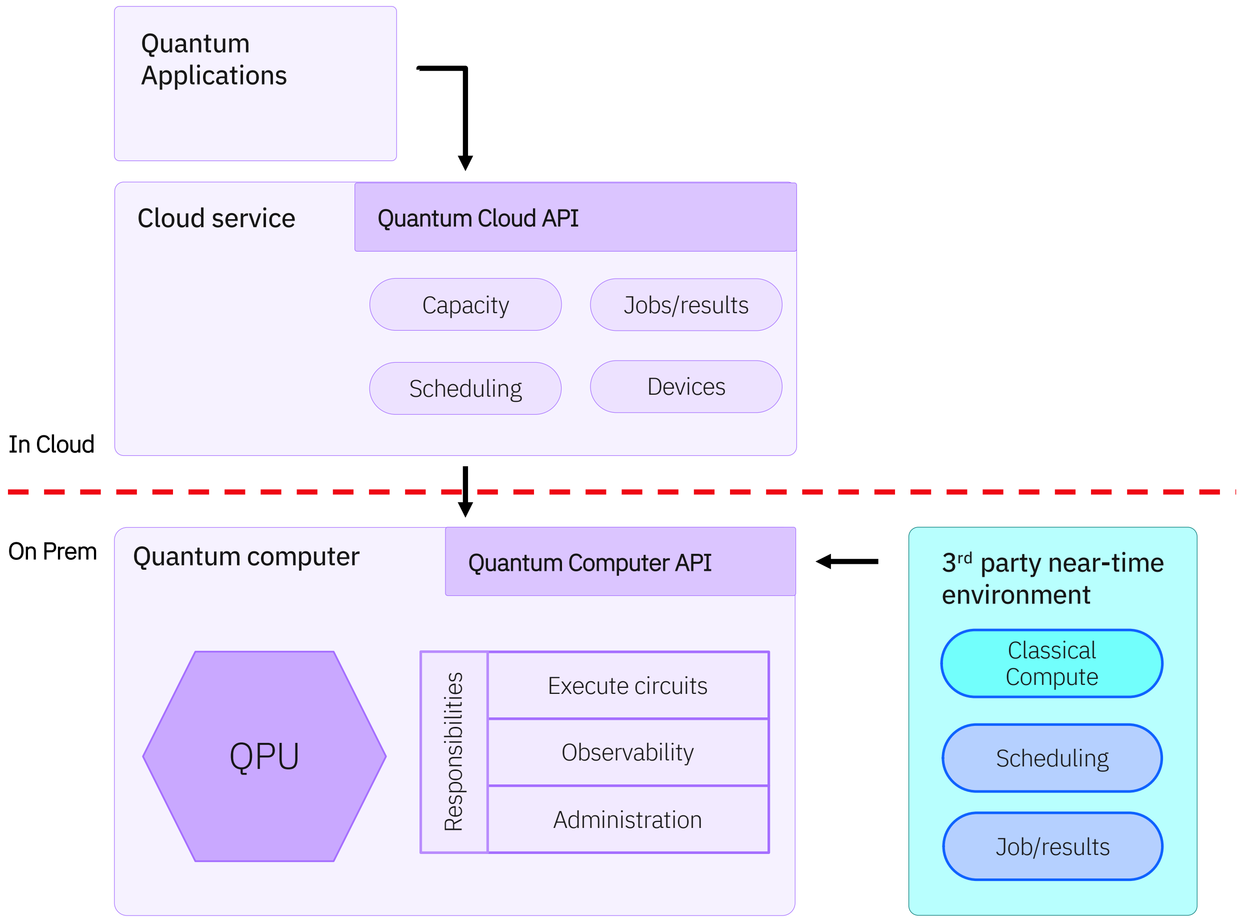
QPU Vs Quantum Computer

Abstract elements

- The ***QPU*** is the set of necessary signal-driven hardware and elements in charge of executing the quantum computation.
- The ***quantum computer*** translates the user input into elements executable by the QPU.
- The Quantum Computer API exposes the quantum computer capabilities and defends against unauthorized access.
- Additional value creation can co-exist in 3rd party near-time environments connected to the quantum computer, or in the cloud.

Responsibilities

- The quantum computer exposes interfaces for executing quantum circuits, observe, and administrate.
- Other environments will implement additional responsibilities on top of the Quantum Computer API.
- I.e.: interfacing environment may implement multi-tenancy responsibilities such as user access or scheduling.



Middleware is used to connect from your bash script to the quantum stack

ORNL laid out the fundamentals for Middleware* to be:

- A unified resource management system that efficiently coordinates quantum and classical resources
- A flexible quantum programming interface that abstracts hardware-specific details
- A Quantum Platform Manager API that simplifies the integration of various quantum hardware systems
- A comprehensive **tool chain for quantum circuit optimization and execution**

***Shehata et al. (2025)** *Bridging Paradigms: Designing for HPC–Quantum Convergence*
Future Generation Computer Systems, Vol. 174
doi.org/10.1016/j.future.2025.107980

What is Slurm

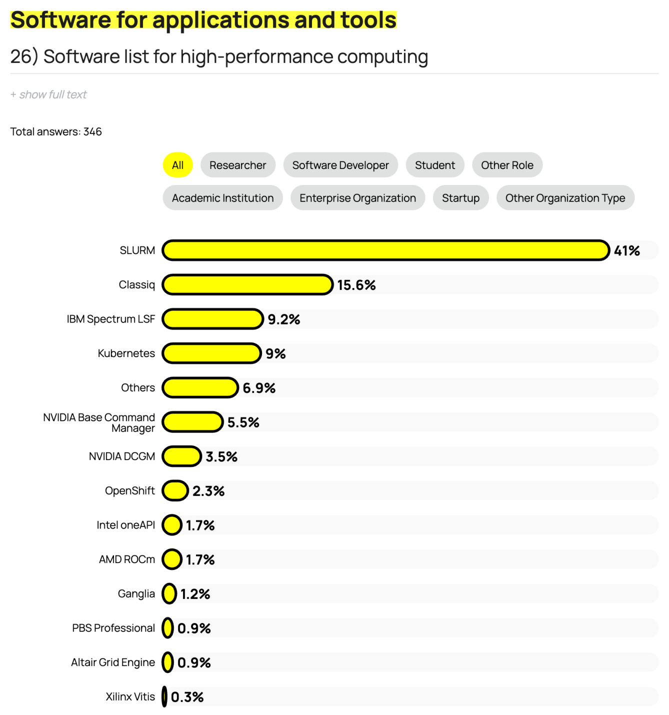
An open-source job scheduler and workload manager for HPC clusters, efficiently allocating resources (CPUs, memory, GPUs) to users' jobs, managing execution, and handling job queues and priorities, making it easier to run large-scale parallel tasks on shared systems.

Key Functions

- **Resource Allocation**: Assigns compute nodes, CPUs, GPUs, and memory to jobs.
- **Job Scheduling**: Orders pending jobs based on priority, submission time, and resource requirements.
- **Job Management**: Handles job submission, monitoring, cancellation, and execution.
- **Environment**: Runs on Linux clusters, requiring no kernel modifications, and is highly scalable.

User Interactions

- **Write a Script**: Create a script (e.g., bash) with directives for resources
- **Submit**: Use *sbatch* to send it to the queue.
- **Monitor**: Use *squeue* to see pending/running jobs or *scontrol show job <JOBID>* for details.
- **Cancel**: Use *scancel <JOBID>* to stop a job.
- **Interactive Jobs**: Use *srun* for immediate access to resources or *salloc* to request an allocation.



Software for applications and tools

26) Software list for high-performance computing

+ show full text

Total answers: 346

2025 Quantum Open Source Survey
https://unitaryfoundation.github.io/survey-2025/#Software

Proposed Quantum Resource Management Solution

Highlights from Our 2025 Activities

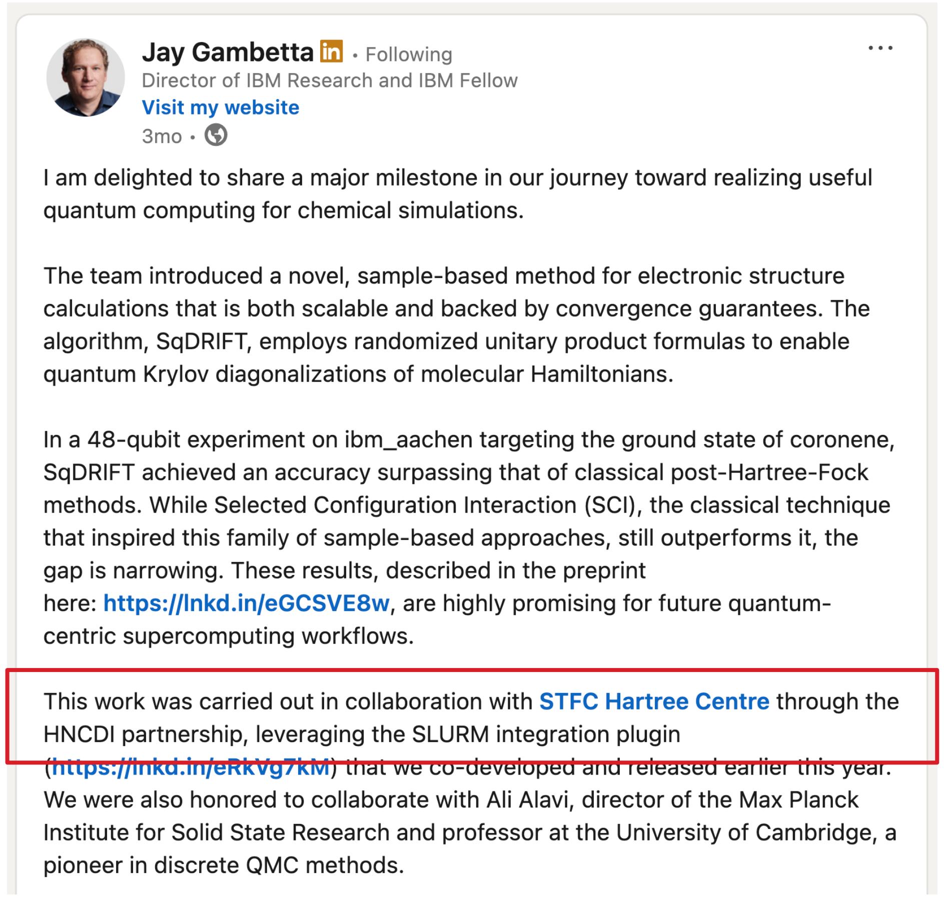
We introduce a unified resource management solution that delivers the Quantum Resource Management Interface (QRMI) to abstract vendor-specific quantum hardware and uses the SPANK plugins framework to control hybrid classical–quantum workflows in Slurm.

- Native QPU integration with Workflow Manager - **Slurm**
- Unified job submission for quantum + classical tasks.
- Extensible and adaptable architecture for other QPU vendors.
- Improved usability, scheduling, and resource control.
- Extensible to cloud-native environments.

- **Fully open-source** stack with easy installation - runnable
- Partnered with **10+ organizations**.
- Deployed to **4+ HPC data centers** (STFC, RPI and others) and started evaluations
- Enabled **partner's experiments for scientific breakthroughs**.
- Continuously enhancing the stack based on partner feedback

From Integration to Innovation: QRMI and Slurm Driving Quantum Research

“The hardware experiments were managed via the quantum spank plugin for Slurm v0.1.0 [70].”



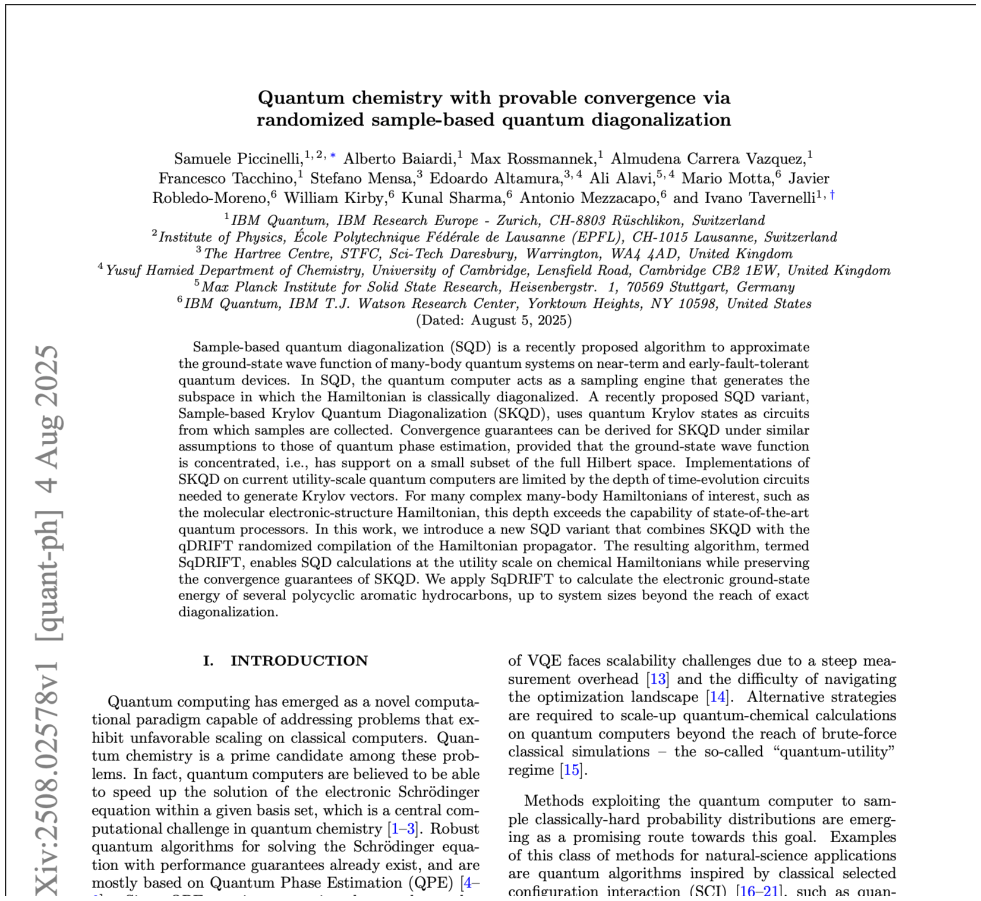
Jay Gambetta · Following
Director of IBM Research and IBM Fellow
Visit my website
3mo ·

I am delighted to share a major milestone in our journey toward realizing useful quantum computing for chemical simulations.

The team introduced a novel, sample-based method for electronic structure calculations that is both scalable and backed by convergence guarantees. The algorithm, SqDRIFT, employs randomized unitary product formulas to enable quantum Krylov diagonalizations of molecular Hamiltonians.

In a 48-qubit experiment on ibm_aachen targeting the ground state of coronene, SqDRIFT achieved an accuracy surpassing that of classical post-Hartree-Fock methods. While Selected Configuration Interaction (SCI), the classical technique that inspired this family of sample-based approaches, still outperforms it, the gap is narrowing. These results, described in the preprint here: **https://lnkd.in/eGCSVE8w**, are highly promising for future quantum-centric supercomputing workflows.

This work was carried out in collaboration with **STFC Hartree Centre** through the HNCDI partnership, leveraging the SLURM integration plugin (**https://lnkd.in/eRkVg7kM**) that we co-developed and released earlier this year. We were also honored to collaborate with Ali Alavi, director of the Max Planck Institute for Solid State Research and professor at the University of Cambridge, a pioneer in discrete QMC methods.



Quantum chemistry with provable convergence via randomized sample-based quantum diagonalization

Samuele Piccinelli,[1, 2, *] Alberto Baiardi,[1] Max Rossmannek,[1] Almudena Carrera Vazquez,[1] Francesco Tacchino,[1] Stefano Mensa,[3] Edoardo Altamura,[3, 4] Ali Alavi,[5, 4] Mario Motta,[6] Javier Robledo-Moreno,[6] William Kirby,[6] Kunal Sharma,[6] Antonio Mezzacapo,[6] and Ivano Tavernelli[1, †]

[1] *IBM Quantum, IBM Research Europe - Zurich, CH-8803 Rüschlikon, Switzerland*
[2] *Institute of Physics, École Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland*
[3] *The Hartree Centre, STFC, Sci-Tech Daresbury, Warrington, WA4 4AD, United Kingdom*
[4] *Yusuf Hamied Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, United Kingdom*
[5] *Max Planck Institute for Solid State Research, Heisenbergstr. 1, 70569 Stuttgart, Germany*
[6] *IBM Quantum, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, United States*

(Dated: August 5, 2025)

Sample-based quantum diagonalization (SQD) is a recently proposed algorithm to approximate the ground-state wave function of many-body quantum systems on near-term and early-fault-tolerant quantum devices. In SQD, the quantum computer acts as a sampling engine that generates the subspace in which the Hamiltonian is classically diagonalized. A recently proposed SQD variant, Sample-based Krylov Quantum Diagonalization (SKQD), uses quantum Krylov states as circuits from which samples are collected. Convergence guarantees can be derived for SKQD under similar assumptions to those of quantum phase estimation, provided that the ground-state wave function is concentrated, i.e., has support on a small subset of the full Hilbert space. Implementations of SKQD on current utility-scale quantum computers are limited by the depth of time-evolution circuits needed to generate Krylov vectors. For many complex many-body Hamiltonians of interest, such as the molecular electronic-structure Hamiltonian, this depth exceeds the capability of state-of-the-art quantum processors. In this work, we introduce a new SQD variant that combines SKQD with the qDRIFT randomized compilation of the Hamiltonian propagator. The resulting algorithm, termed SqDRIFT, enables SQD calculations at the utility scale on chemical Hamiltonians while preserving the convergence guarantees of SKQD. We apply SqDRIFT to calculate the electronic ground-state energy of several polycyclic aromatic hydrocarbons, up to system sizes beyond the reach of exact diagonalization.

I. INTRODUCTION

Quantum computing has emerged as a novel computational paradigm capable of addressing problems that exhibit unfavorable scaling on classical computers. Quantum chemistry is a prime candidate among these problems. In fact, quantum computers are believed to be able to speed up the solution of the electronic Schrödinger equation within a given basis set, which is a central computational challenge in quantum chemistry [1–3]. Robust quantum algorithms for solving the Schrödinger equation with performance guarantees already exist, and are mostly based on Quantum Phase Estimation (QPE) [4–

of VQE faces scalability challenges due to a steep measurement overhead [13] and the difficulty of navigating the optimization landscape [14]. Alternative strategies are required to scale-up quantum-chemical calculations on quantum computers beyond the reach of brute-force classical simulations – the so-called “quantum-utility” regime [15].

Methods exploiting the quantum computer to sample classically-hard probability distributions are emerging as a promising route towards this goal. Examples of this class of methods for natural-science applications are quantum algorithms inspired by classical selected configuration interaction (SCI) [16–21], such as quan-

Component Roles & Responsibilities

Slurm
HPC Resource Manager

- Providing unified job submission for quantum + classical tasks
- Job Scheduling - fair share, priority, and partition policies
- Generic resource allocation - heterogenous resources and accelerators
- Admin and security - secrets, resource security and allocation, user authentication

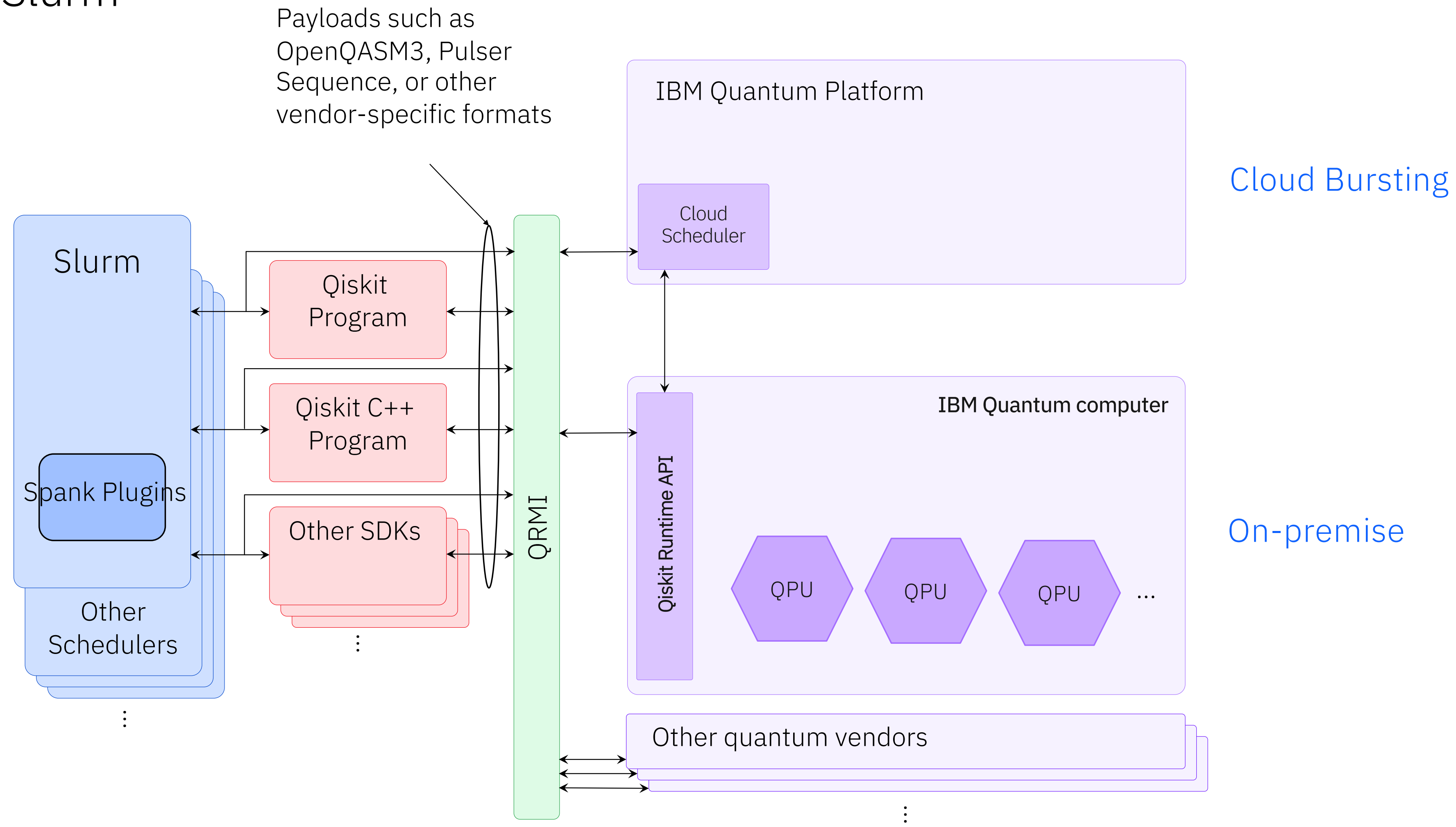
SPANK Plugin
Bridge the gap to QPU resources

- (Slurm Plug-in Architecture for Node and job Kontrol)
- Exposes QPUs as generic resource to Slurm
- Gives Slurm access to the tools to check accessibility, acquire the resources required
- Passes environment variables for connection to QPU resources

QRMI
Manage Quantum Resource

- Accesses resource through API or direct access - customizable
- Acquire the resource for the length of script
- Abstract beyond device specifications and passes any device specific options defined
- Create and monitor quantum jobs

QRMI / Slurm


Payloads such as OpenQASM3, Pulser Sequence, or other vendor-specific formats
Slurm
Spank Plugins
Other Schedulers
Qiskit Program
Qiskit C++ Program
Other SDKs
QRMI
IBM Quantum Platform
Cloud Scheduler
Cloud Bursting
IBM Quantum computer
Qiskit Runtime API
QPU
QPU
QPU
...
On-premise
Other quantum vendors

Slurm User Experience

HPC user: use slurm quantum resource

backend is selected in the *slurm resource definition* used for jobs, e.g. *ibm_fez*

resource defined in slurm parameters (configuration)

```
#SBATCH --time=100
#SBATCH --output=<LOGS_PATH>
#SBATCH --gres=qpu:1
#SBATCH --qpu=ibm_fez
#SBATCH --... # other options

srun ...
```

slurm resource is picked up by *application source code* and used for transpilation and execution:

use previously defined resource

```
from qiskit import QuantumCircuit
from qrmi_primitives import QRMIService
# using an IBM QRMI flavor:
from qrmi_primitives.ibm import SamplerV2

# define circuit
circuit = ...

# inject credentials needed for accessing the service at this point
load_dotenv()

# instantiate QRMI service and get quantum resource
service = QRMIService()

resources = service.resources()
qrmi = resources[0]

# Generate transpiler target and transpile
target = get_target(qrmi)
pm = generate_preset_pass_manager(
    optimization_level=1,
    target=target,
)
isa_circuit = pm.run(circuit)

# run the circuit
sampler = SamplerV2(qrmi, options={})
```

HPC admin: define access to physical quantum resources

slurm quantum plugin configuration defines which resources are made available for use (e.g. *ibm_fez*)

```
{
  "resources": [
    {
      "name": ”ibm_fez",
      "type": "qiskit-runtime-service",
      "environment": {
        "QRMI_IBM_QRS_ENDPOINT": "https://quantum.cloud.ibm.com/api/v1",
        "QRMI_IBM_QRS_IAM_ENDPOINT": "https://iam.cloud.ibm.com",
        "QRMI_IBM_QRS_IAM_APIKEY": "...",
        "QRMI_IBM_QRS_SERVICE_CRN": "..."
      }
    },
    {
      "name": "FRESNEL",
      "type": "pasqal-cloud",
      "environment": {
        "QRMI_PASQAL_CLOUD_PROJECT_ID": "...",
        "QRMI_PASQAL_CLOUD_AUTH_TOKEN": "..."
      }
    }
  ]
}
```

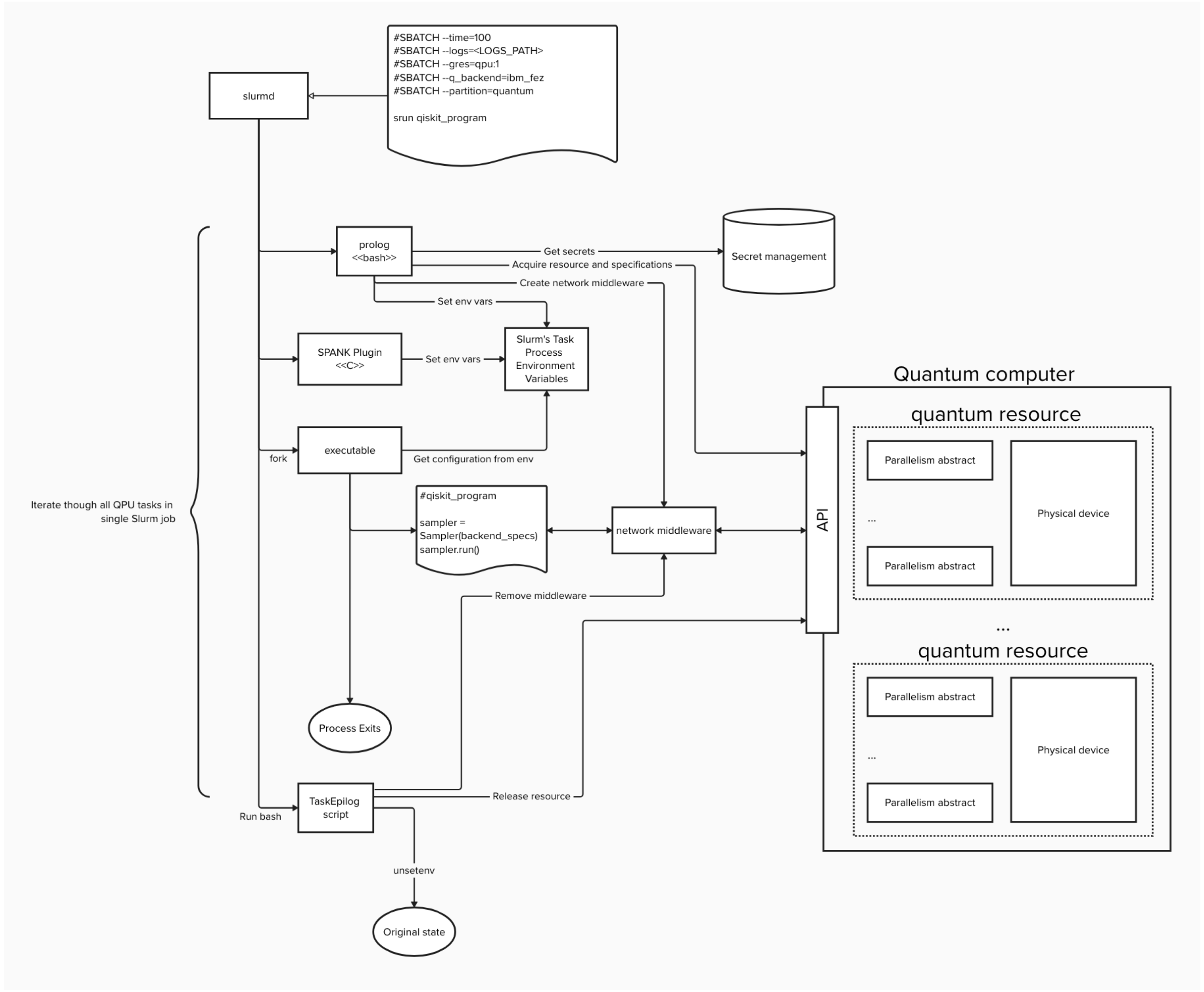
In slurm.conf, qpu generic resources can be made available to nodes:

```
...
GresTypes=qpu,name
NodeName=node[1-5000] Gres=qpu,name:ibm_fez
...
```

See ux documentation on github for more details
https://github.com/qiskit-community/spank-plugins/blob/main/docs/ux.md

General Architecture of Plugin

- Lifecycle for Quantum Jobs
  - Prologue: Acquire resource, setup middleware
  - Task Init: Setup env
  - Epilogue: Release resource, cleanup
- Natural integration with Slurm job lifecycle
  - Enables quantum-classical workflows using familiar HPC tooling



```
#SBATCH --time=100
#SBATCH --logs=<LOGS_PATH>
#SBATCH --gres=qpu:1
#SBATCH --q_backend=ibm_fez
#SBATCH --partition=quantum

srun qiskit_program
```

```
#qiskit_program

sampler =
Sampler(backend_specs)
sampler.run()
```

Environment Variables by Spank Plugin

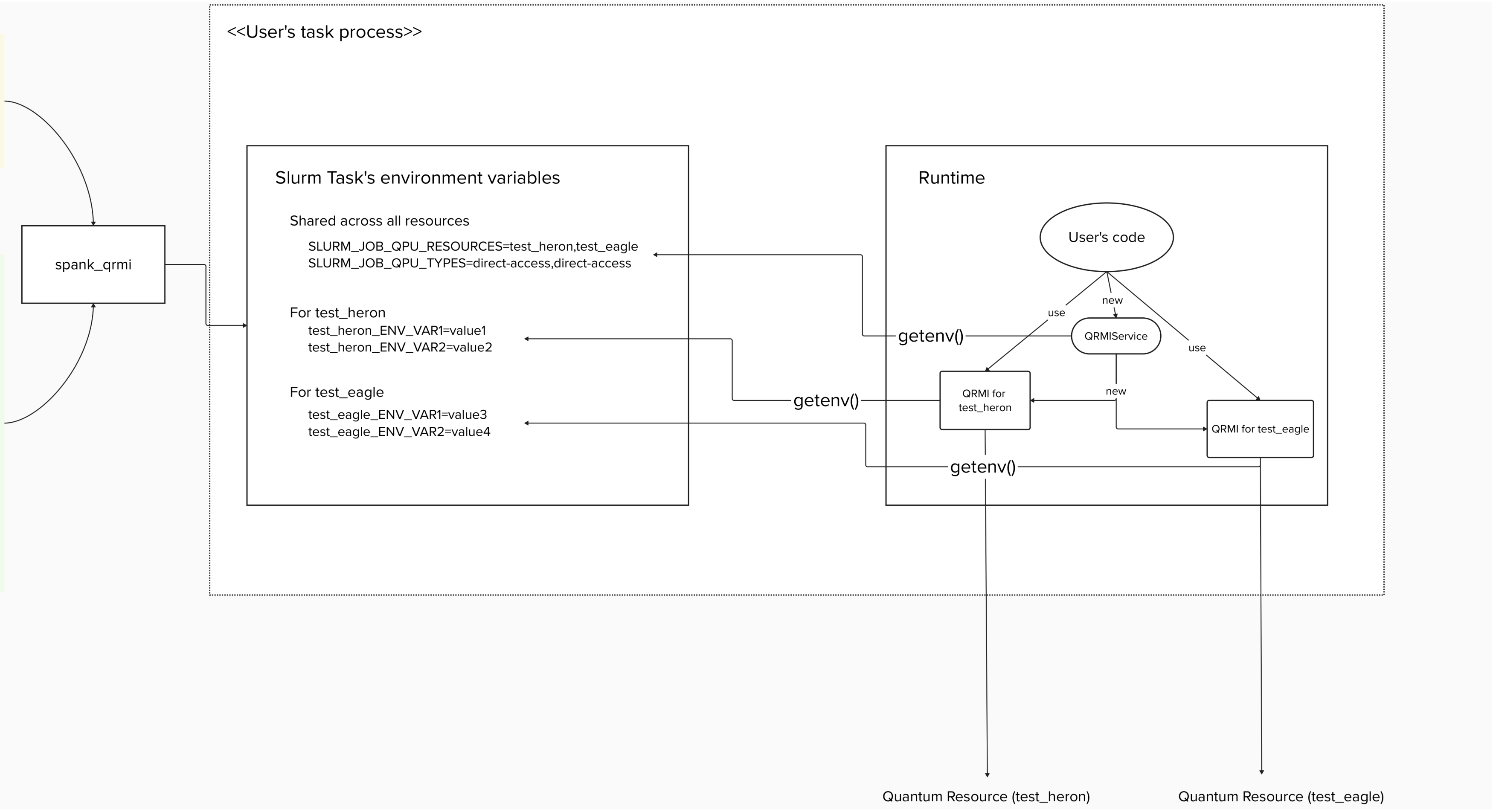
Job script

```bash
#!/bin/bash
#SBATCH --job-name=sampler_job
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=1
#SBATCH --qpu=test_heron,test_eagle

source /shared/pyenv/bin/activate
srun python /shared/job_scripts/sampler.py
```

qrmi_config.json

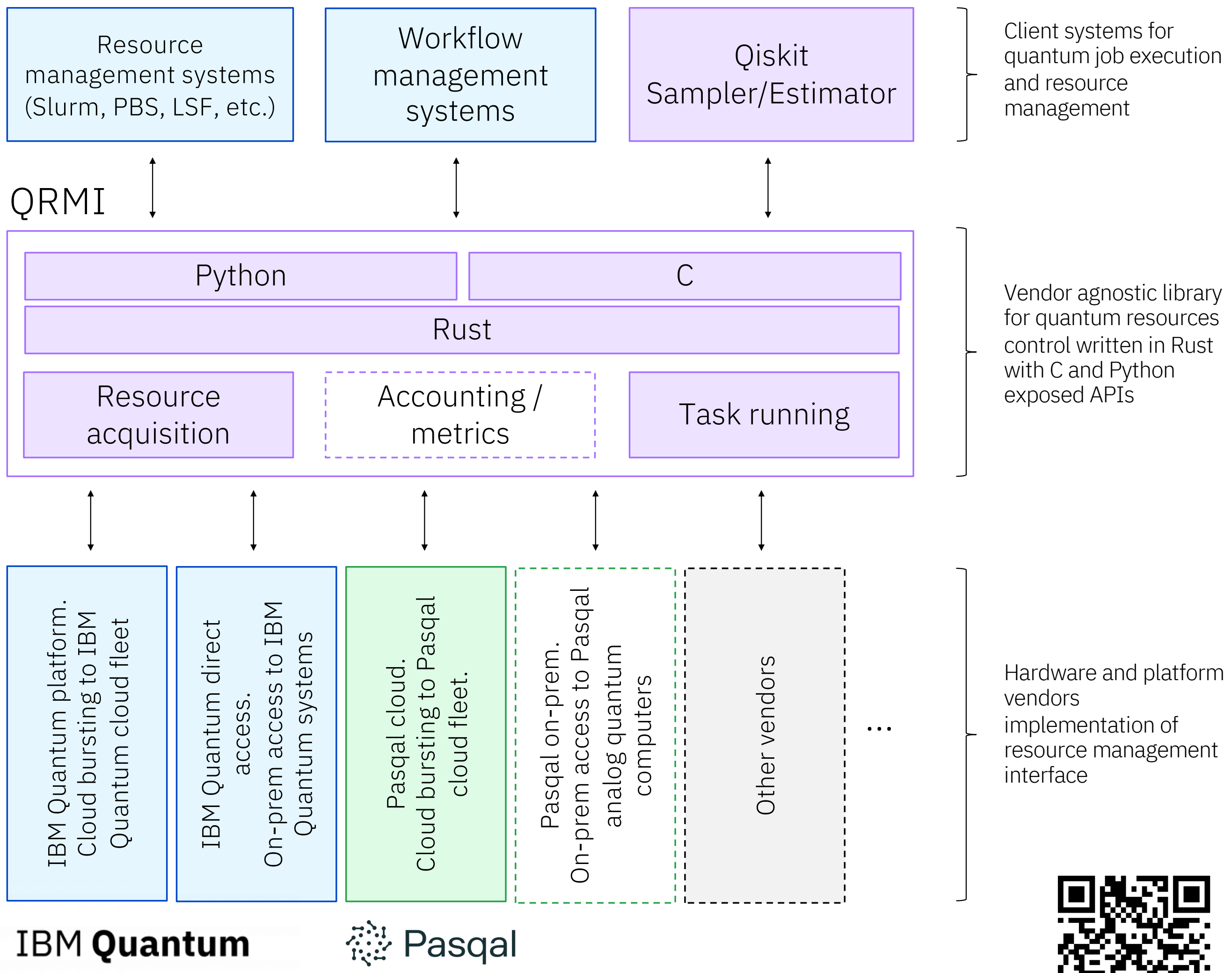
```json
{
  "resources": [
   {
    "name": "test_heron",
    "type": "direct-access",
    "environment": {
      "ENV_VAR1": "value1",
      "ENV_VAR2": "value2",
    }
   },
   {
    "name": "test_eagle",
    "type": "direct-access",
    "environment": {
       "ENV_VAR1": "value3",
       "ENV_VAR2": "value4"
    }
   }
  ]
}
```



spank_qrmi
<<User's task process>>
Slurm Task's environment variables
Shared across all resources
SLURM_JOB_QPU_RESOURCES=test_heron,test_eagle
SLURM_JOB_QPU_TYPES=direct-access,direct-access
For test_heron
test_heron_ENV_VAR1=value1
test_heron_ENV_VAR2=value2
For test_eagle
test_eagle_ENV_VAR1=value3
test_eagle_ENV_VAR2=value4
Runtime
User's code
use
new
use
QRMIService
new
getenv()
getenv()
getenv()
QRMI for test_heron
QRMI for test_eagle
Quantum Resource (test_heron)
Quantum Resource (test_eagle)

Quantum Resource Management Interface (QRMI)

- Thin and vendor agnostic layer to access, control and monitor underlying on-prem or cloud Quantum systems
- Exposes notion of quantum resources, which might be qubit, entire physical system, virtual QPU, etc.
- 3 sets of APIs available for QRMI:
  - Resource acquisition: availability, acquire, release.
  - Task running: execution on quantum payload on target hardware
  - [2026 1H] Accounting/metrics: system utilization, etc.
- Provides loose coupling between clients and systems


Resource management systems (Slurm, PBS, LSF, etc.)
Workflow management systems
Qiskit Sampler/Estimator
Client systems for quantum job execution and resource management
QRMI
Python
C
Rust
Resource acquisition
Accounting / metrics
Task running
Vendor agnostic library for quantum resources control written in Rust with C and Python exposed APIs
IBM Quantum platform. Cloud bursting to IBM Quantum cloud fleet
IBM Quantum direct access. On-prem access to IBM Quantum systems
Pasqal cloud. Cloud bursting to Pasqal cloud fleet.
Pasqal on-prem. On-prem access to Pasqal analog quantum computers
Other vendors
…
Hardware and platform vendors implementation of resource management interface
IBM Quantum
Pasqal

QuantumResource

- Defines interfaces to quantum resources. Each Quantum vendor must implement this Rust trait.
- https://github.com/qiskit-community/qrmi/blob/main/src/lib.rs

▨ also used by Slurm plugin

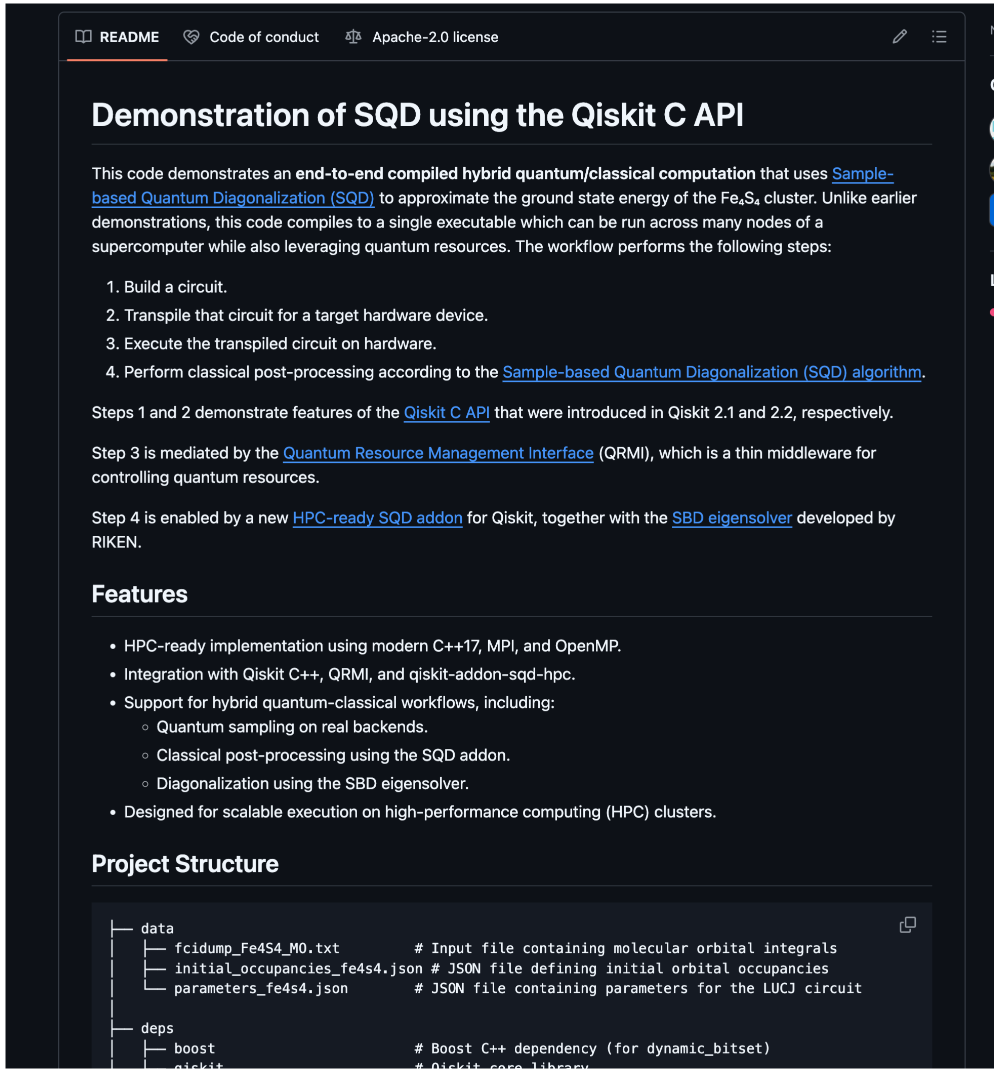
Function	Descriptions
is_accessible	Returns true if device is accessible, otherwise false.
acquire	Acquires quantum resource and returns acquisition token if succeeded. If no one owns the lock, it acquires the lock and returns immediately. If another owns the lock, block until we are able to acquire lock.
release	Releases quantum resource
task_start	Start a task and returns an identifier of this task if succeeded.
task_stop	Stops the task specified by `task_id`. This function is called if the user cancels the job or if the time limit for job execution is exceeded. The implementation must cancel the task if it is still running.
task_status	Returns the current status of the task specified by `task_id`.
task_result	Returns the results of the task.
task_logs	Returns the log messages of the task.
target	Returns a Target for the specified device. Vendor specific serialized data. This might contain the constraints(instructions, properties and timing information etc.) of a particular device to allow compilers to compile an input circuit to something that works and is optimized for a device.
metadata	Returns other specific to system or device data

Demo

- Demo environment, Setup
- Unified user experiences to manage Quantum job submission
  - Prepare quantum workload
    - Porting Qiskit getting-started code
  - Create script to describe Slurm job
  - Submit job and monitor job status
  - Review output
- Quantum-MPI hybrid job

https://github.com/ohtanim/SCA-HPCAsia-2026

Enabling More Complex Hybrid Quantum-Classical Computation Workloads with Qiskit C++, QRMI and Slurm



Demonstration of SQD using the Qiskit C API

This code demonstrates an **end-to-end compiled hybrid quantum/classical computation** that uses Sample-based Quantum Diagonalization (SQD) to approximate the ground state energy of the Fe_4S_4 cluster. Unlike earlier demonstrations, this code compiles to a single executable which can be run across many nodes of a supercomputer while also leveraging quantum resources. The workflow performs the following steps:

1. Build a circuit.
2. Transpile that circuit for a target hardware device.
3. Execute the transpiled circuit on hardware.
4. Perform classical post-processing according to the Sample-based Quantum Diagonalization (SQD) algorithm.

Steps 1 and 2 demonstrate features of the Qiskit C API that were introduced in Qiskit 2.1 and 2.2, respectively.

Step 3 is mediated by the Quantum Resource Management Interface (QRMI), which is a thin middleware for controlling quantum resources.

Step 4 is enabled by a new HPC-ready SQD addon for Qiskit, together with the SBD eigensolver developed by RIKEN.

Features

- HPC-ready implementation using modern C++17, MPI, and OpenMP.
- Integration with Qiskit C++, QRMI, and qiskit-addon-sqd-hpc.
- Support for hybrid quantum-classical workflows, including:
  - Quantum sampling on real backends.
  - Classical post-processing using the SQD addon.
  - Diagonalization using the SBD eigensolver.
- Designed for scalable execution on high-performance computing (HPC) clusters.

Project Structure

```
├── data
│   ├── fcidump_Fe4S4_MO.txt         # Input file containing molecular orbital integrals
│   ├── initial_occupancies_fe4s4.json # JSON file defining initial orbital occupancies
│   └── parameters_fe4s4.json        # JSON file containing parameters for the LUCJ circuit
│
├── deps
│   ├── boost                        # Boost C++ dependency (for dynamic_bitset)
│   ├── qiskit                       # Qiskit core library
```

QRMI Provider Development Steps

1. Your first step is to explore how you can enable HPC users to leverage your quantum resources effectively.

2. Decide resource name

- Resource names we have today:
  - *“direct-access”*
  - *“qiskit-runtime-service”*
  - *“pasqal-cloud”*

3. Implement API Client to access your quantum backend

- Code commit to dependencies/<resource_name>

4. Implement QuantumResource

- Define Payload Data Format (input to Quantum resource)
  - *qrmi_payload_v1_schema.json*
  - *src/models/payload.rs*
- Implement src/<vendor_name>/
- Create QRMI API Examples and test your code

5. Implement user libraries to allow HPC users to use your QRMI

- Python ?
- Code commit to *python/qrmi/*
- Create examples and commit to *examples/*

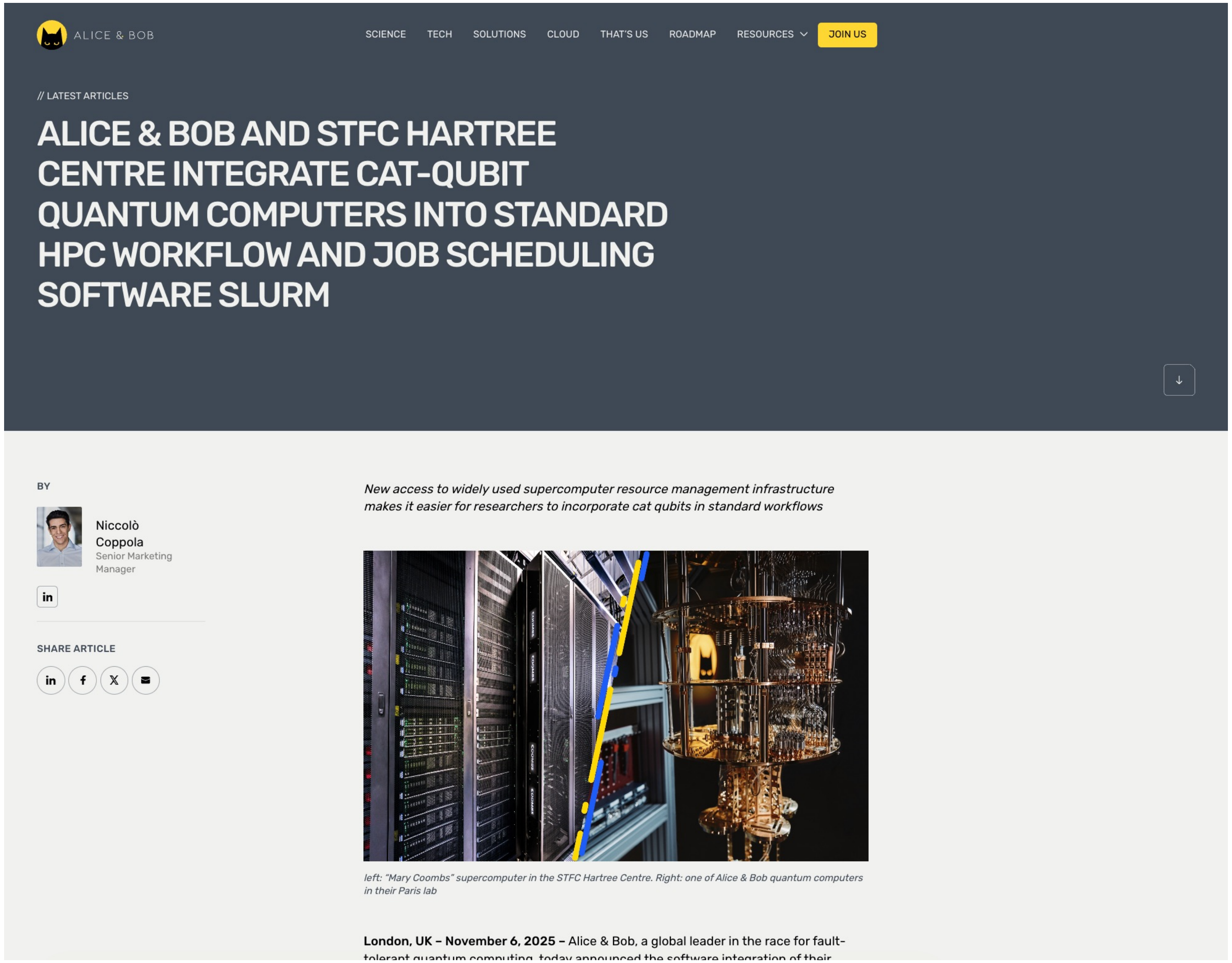
Next Steps

- Rethink Job Scheduling and Resource Acquisitions
- Collaborate closely with the HPC community to understand real-world requirements and continuously deliver requested features
- Expand usage across HPC sites, quantum vendors, and data centers
- Enable interoperability with multiple resource managers
- Package assets: slides, training materials, and self-service hands-on labs
- Harden the codebase for production HPC environments

Requested features

- Accounting Information Support - QPU Usage & Resource utilization
- Network Middleware for security
- "--constraint" option
- Slurm native support
- Enhance logging for observability and troubleshooting

A New QRMI from Alice & Bob is Nearly Ready



// LATEST ARTICLES

ALICE & BOB AND STFC HARTREE CENTRE INTEGRATE CAT-QUBIT QUANTUM COMPUTERS INTO STANDARD HPC WORKFLOW AND JOB SCHEDULING SOFTWARE SLURM

BY

Niccolò Coppola
Senior Marketing Manager

SHARE ARTICLE

New access to widely used supercomputer resource management infrastructure makes it easier for researchers to incorporate cat qubits in standard workflows

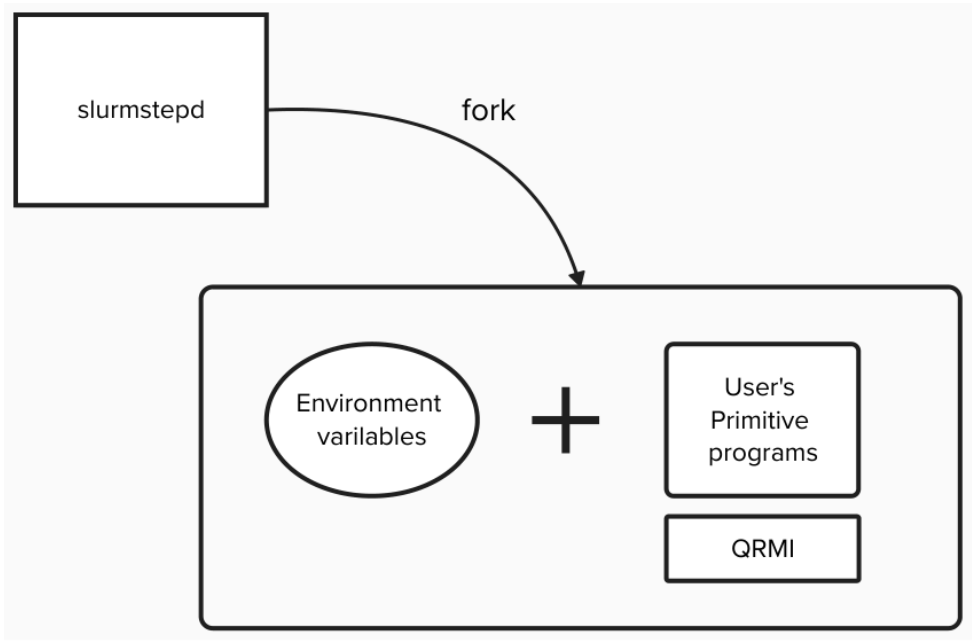
left: "Mary Coombs" supercomputer in the STFC Hartree Centre. Right: one of Alice & Bob quantum computers in their Paris lab

London, UK – November 6, 2025 – Alice & Bob, a global leader in the race for fault-tolerant quantum computing, today announced the software integration of their

https://alice-bob.com/newsroom/hartree-centre-and-alice-bob-integrate-cat-qubits/

IBM Quantum

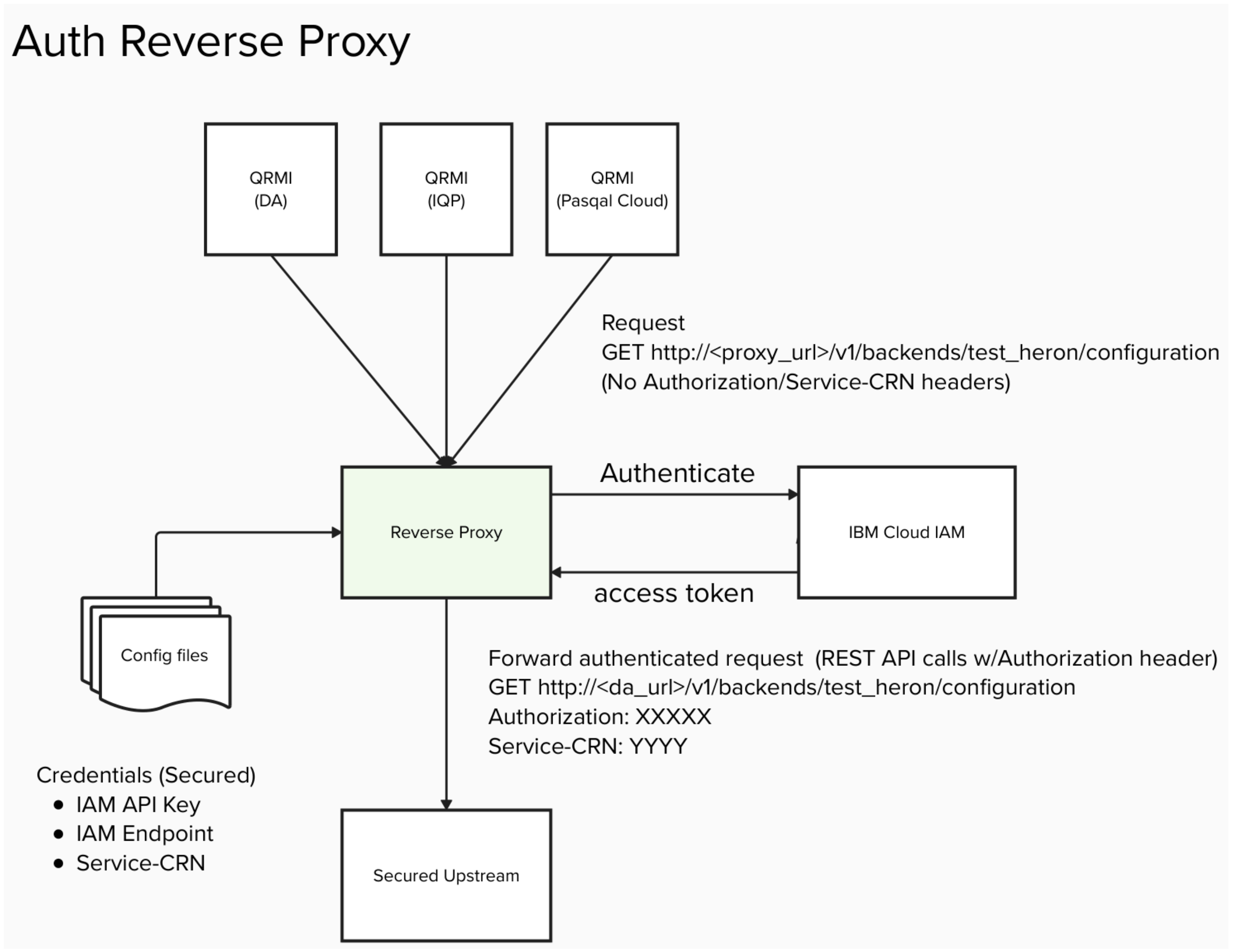
Security issues



- Configuration parameters including credentials are specified via environment variables
- User can know the cluster wide credentials by retrieving all environment variables from code.

Network middleware to resolve the security issues

- Implement as Reverse Proxy, which is separated from Slurm components.
- QRMI clients will submit request without Authorization header.
- Reserve Proxy can access secured credentials and get access token from IAM
- Reverse Proxy will forward the HTTP requests with adding Authorization and other headers



Summary

- Quantum-HPC integration is a critical need for data centers, users and vendors, and remains a key enabler for the near-term success of quantum computers

- This integration introduces multiple challenges – one of the most significant being middleware, which must support multiple vendors, resource allocation, and resource management in heterogeneous environment

- QRMI + SPANK Plugin provides *“a unified resource management solution that delivers the Quantum Resource Management Interface (QRMI) to abstract vendor-specific quantum hardware, and leverages the SPANK plugins framework to control hybrid classical–quantum workflows in Slurm.”*

- We aim to continue driving open, community-driven development, contributing software that empowers HPC partners, accelerates research, and enables strong scientific outcomes.

Open Access to All Our Implementations

QRMI

Thin and vendor agnostic layer to access, control and monitor underlying on-premise or cloud quantum computers

SPANK Plugins

Slurm plugins for Quantum resources and jobs support

Also contains architecture documents (docs/)