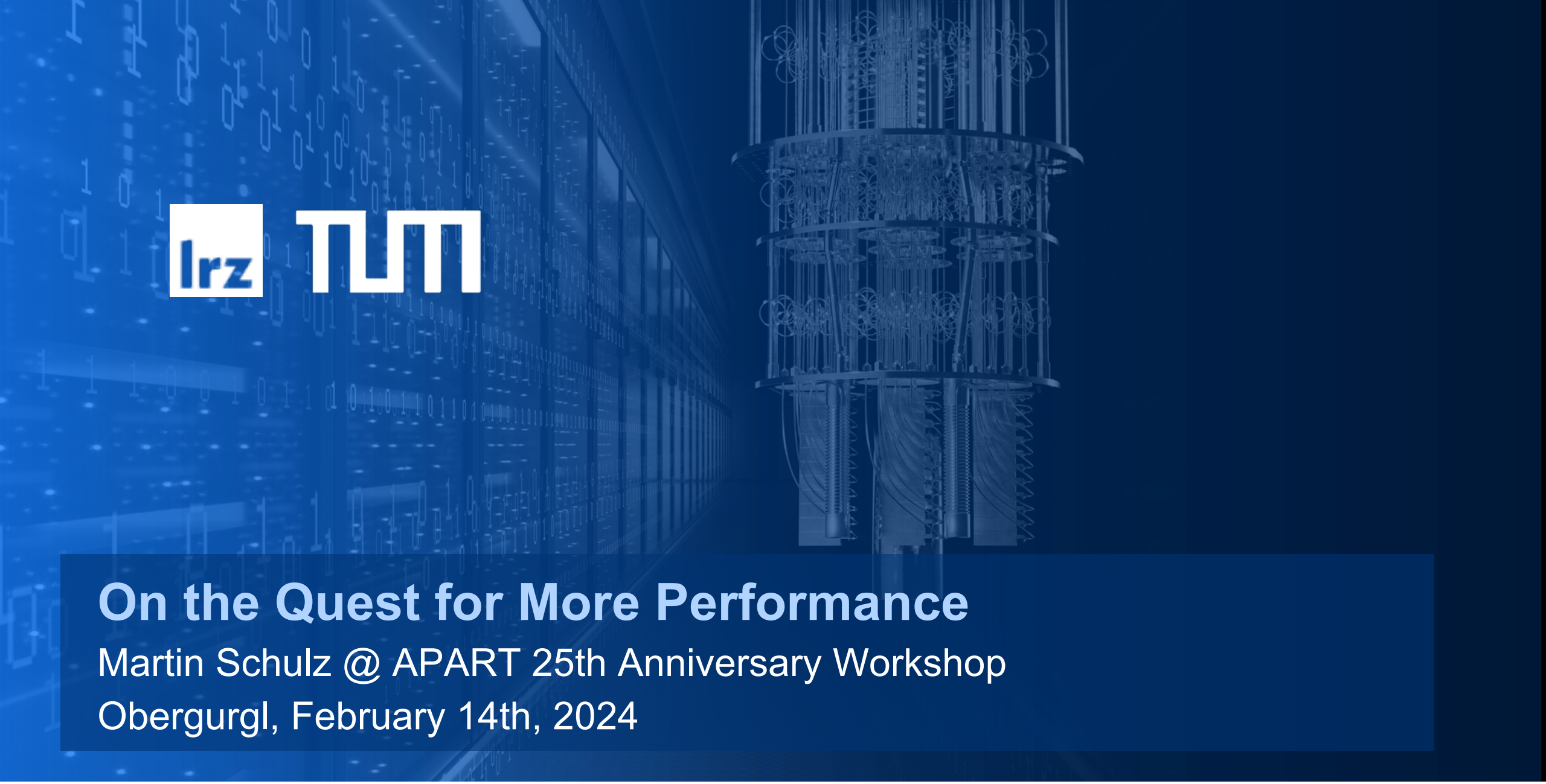


On the Quest for More Performance

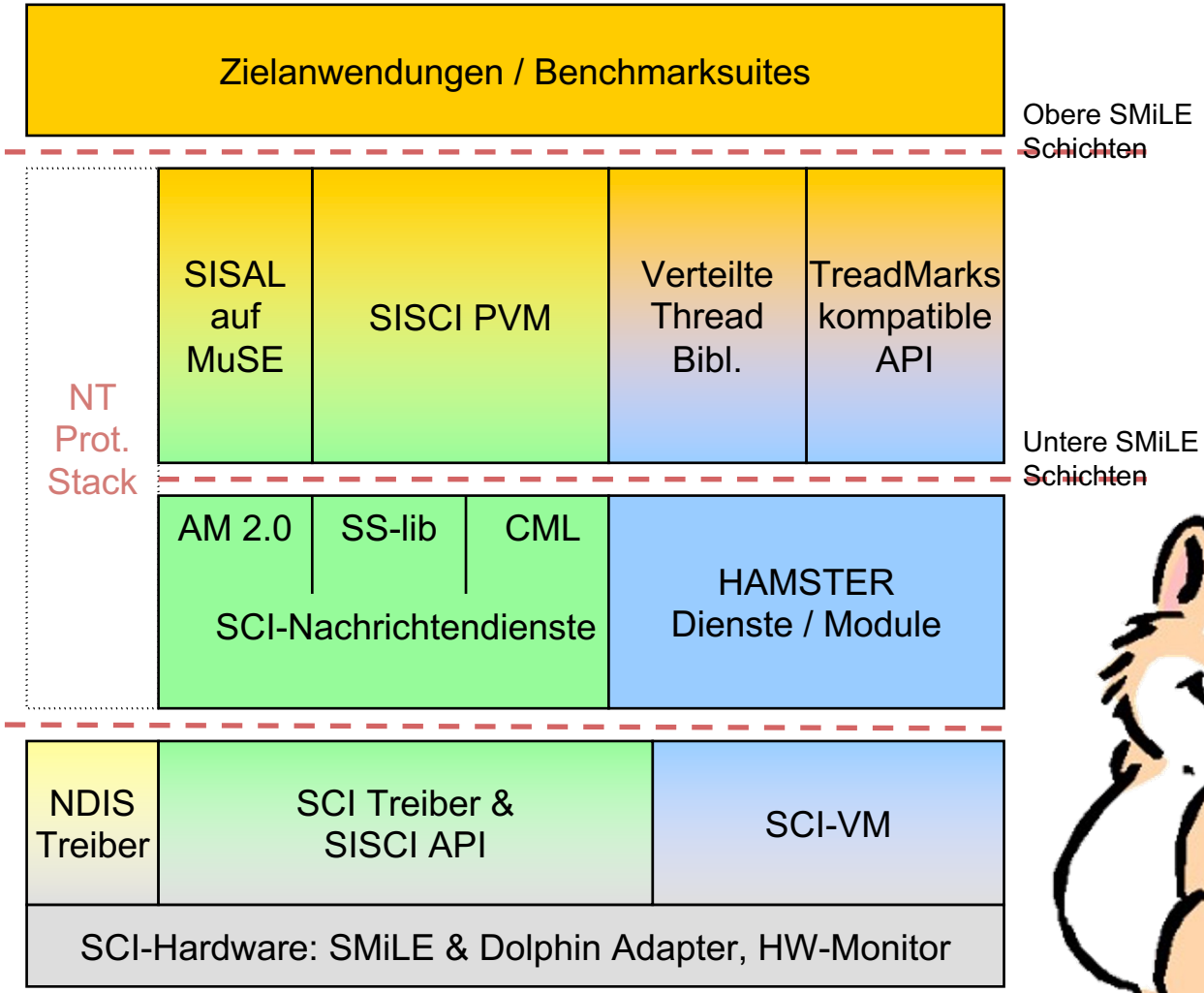
Martin Schulz @ APART 25th Anniversary Workshop

Obergurgl, February 14th, 2024

The Early Days

Performance Engineering für SCI

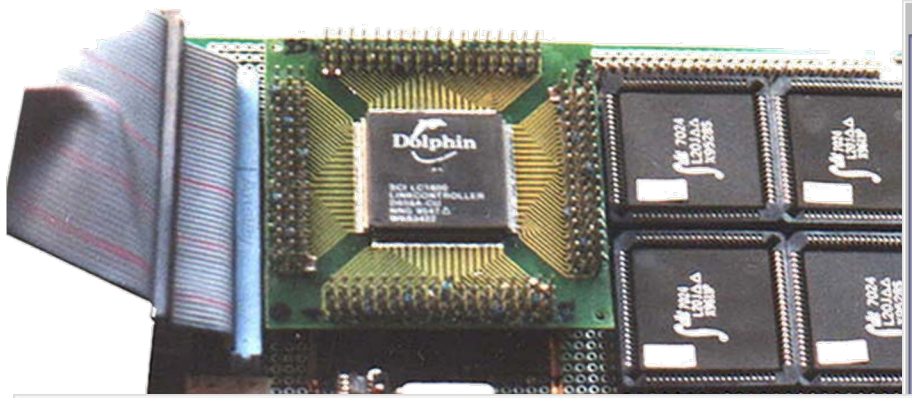
SMiLE

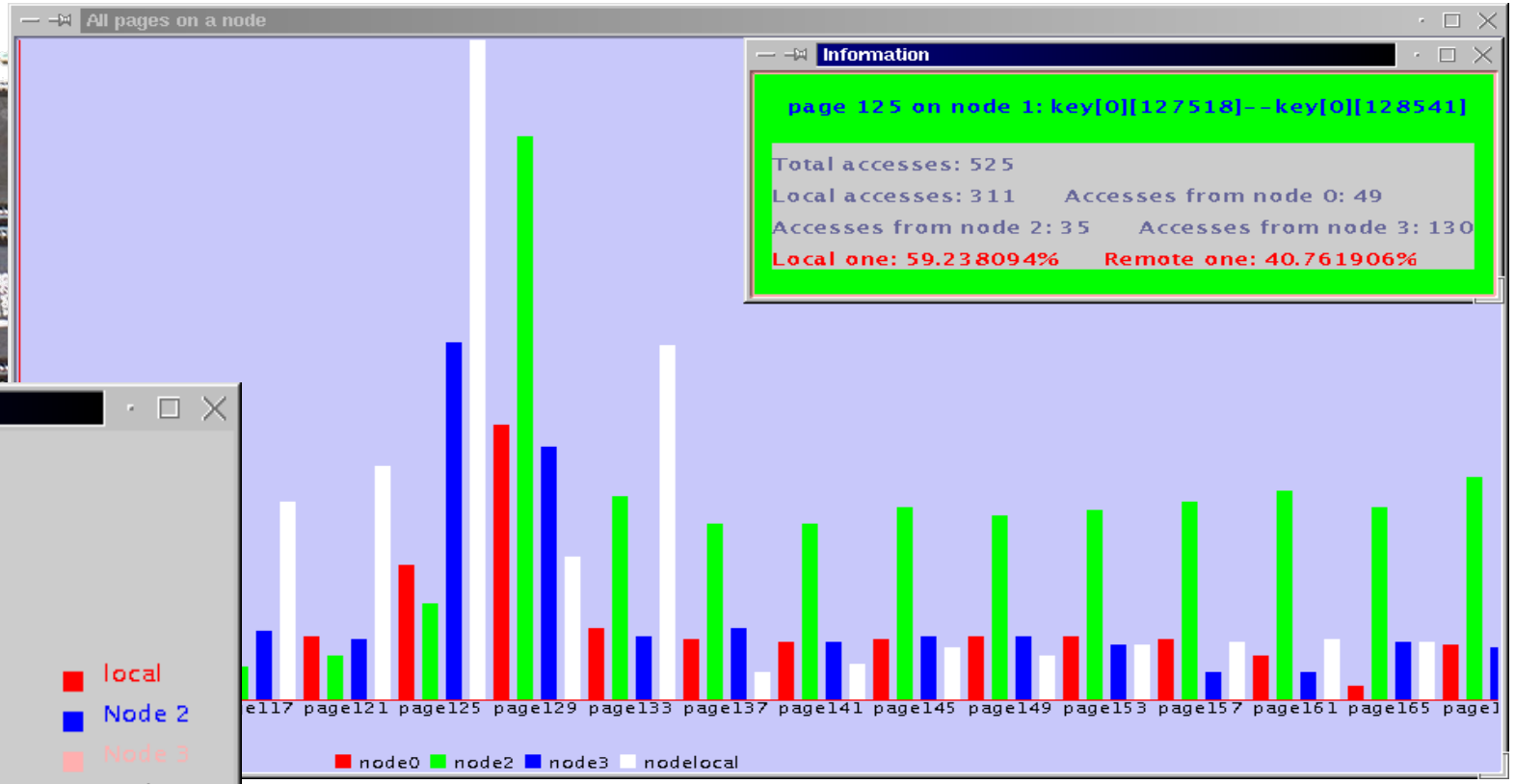


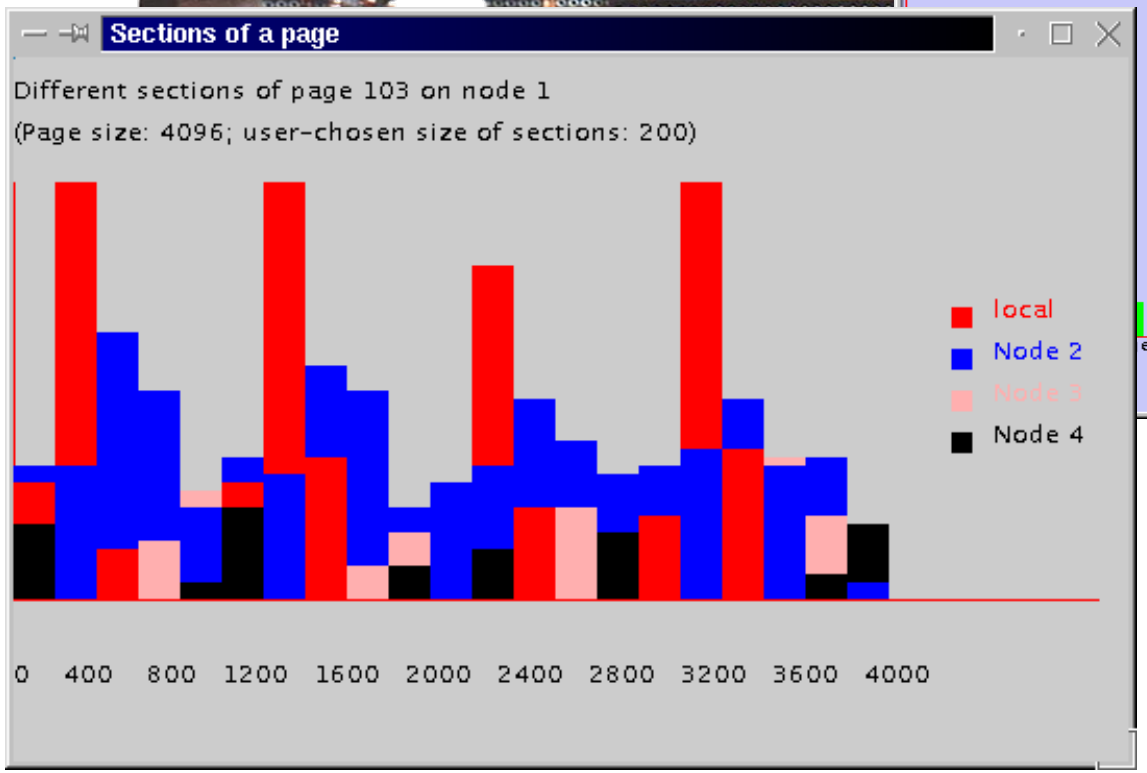
The Early Days

Hardware Monitoring and Tools

SMiLE

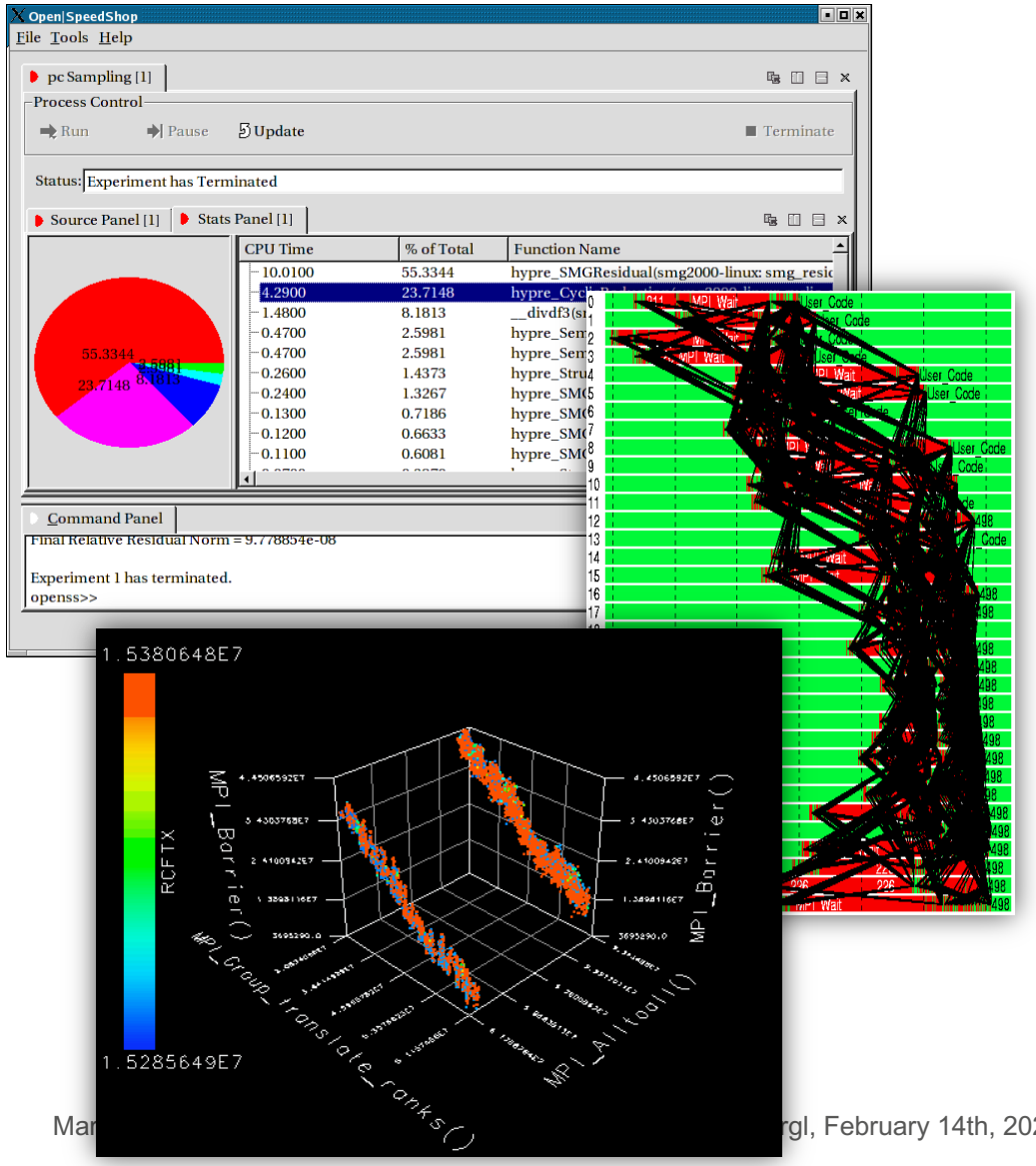


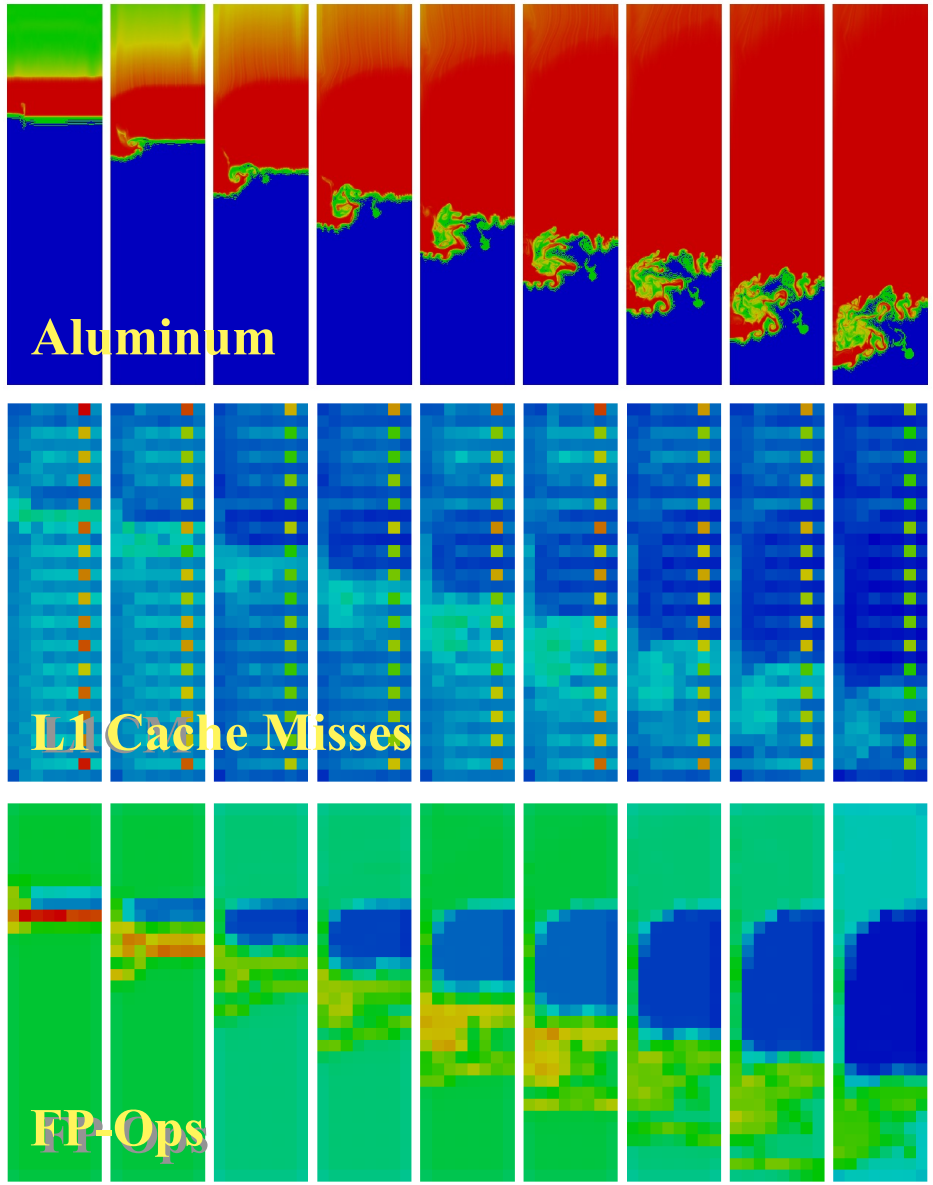




Tools Work at LLNL

From Open|SpeedShop to PAVE



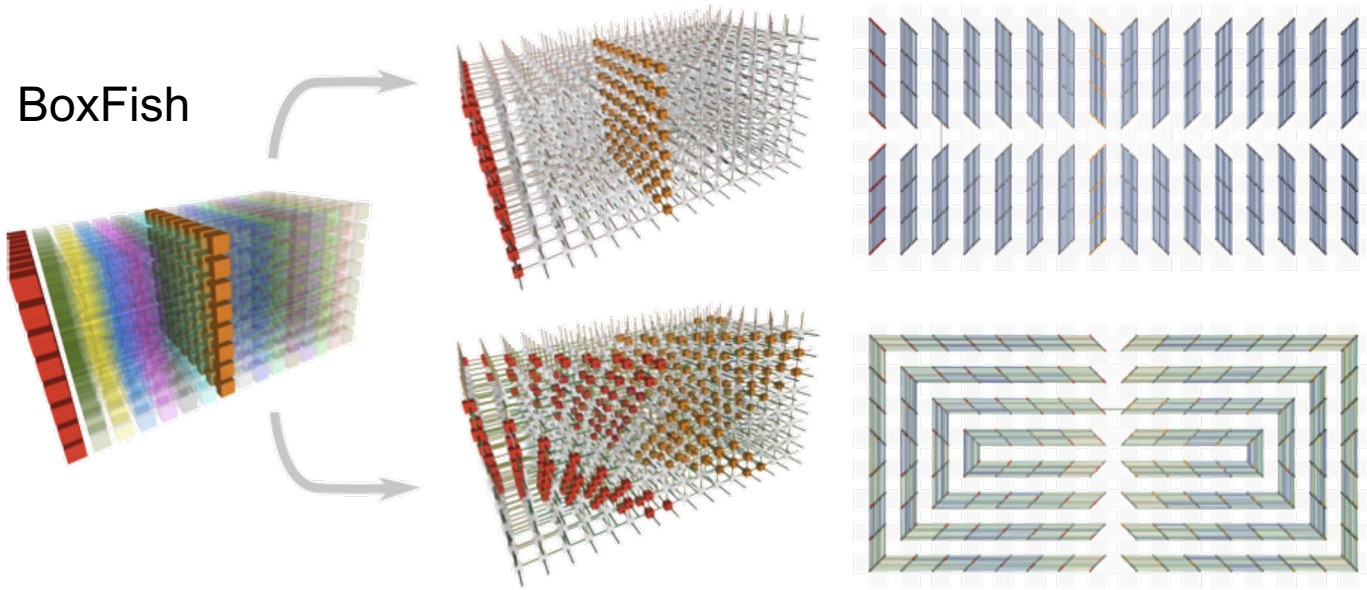


Visual Performance Analysis

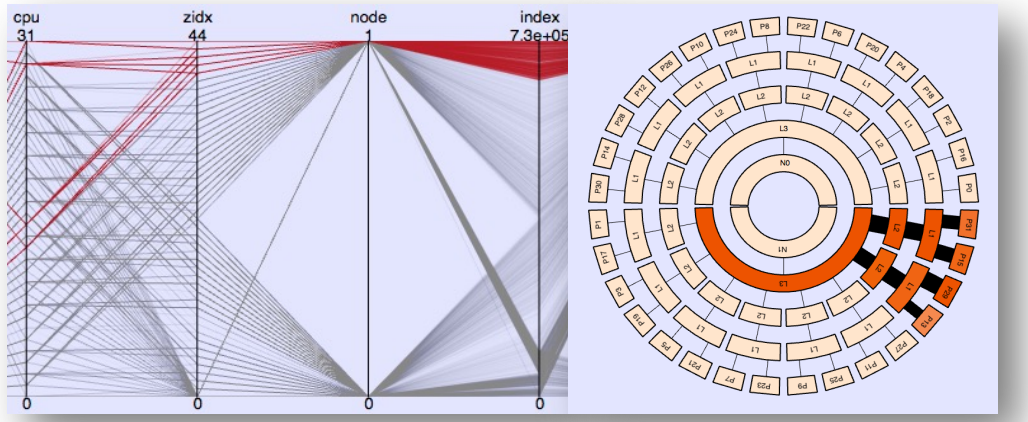
PAVE Tools

Lawrence Livermore National Laboratory

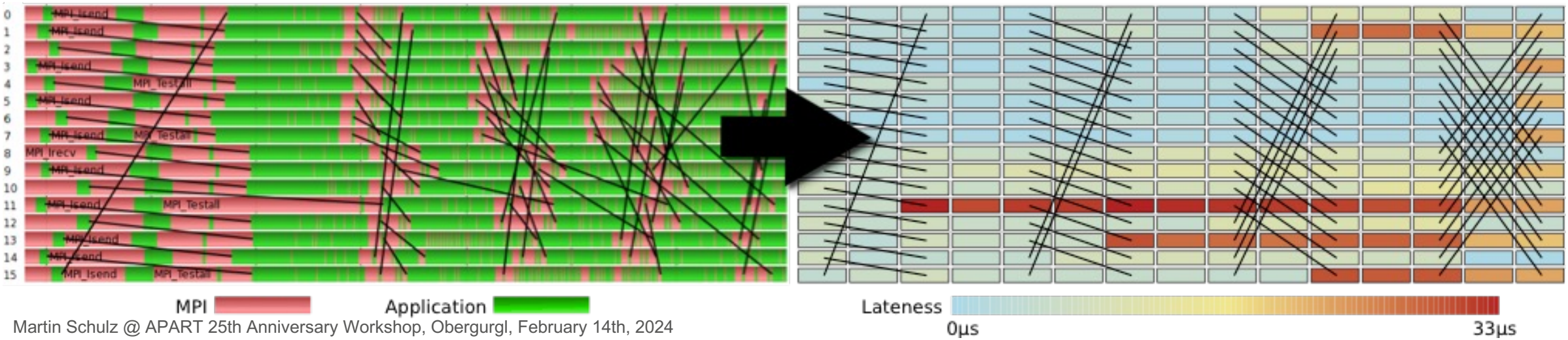
BoxFish



MemAxes



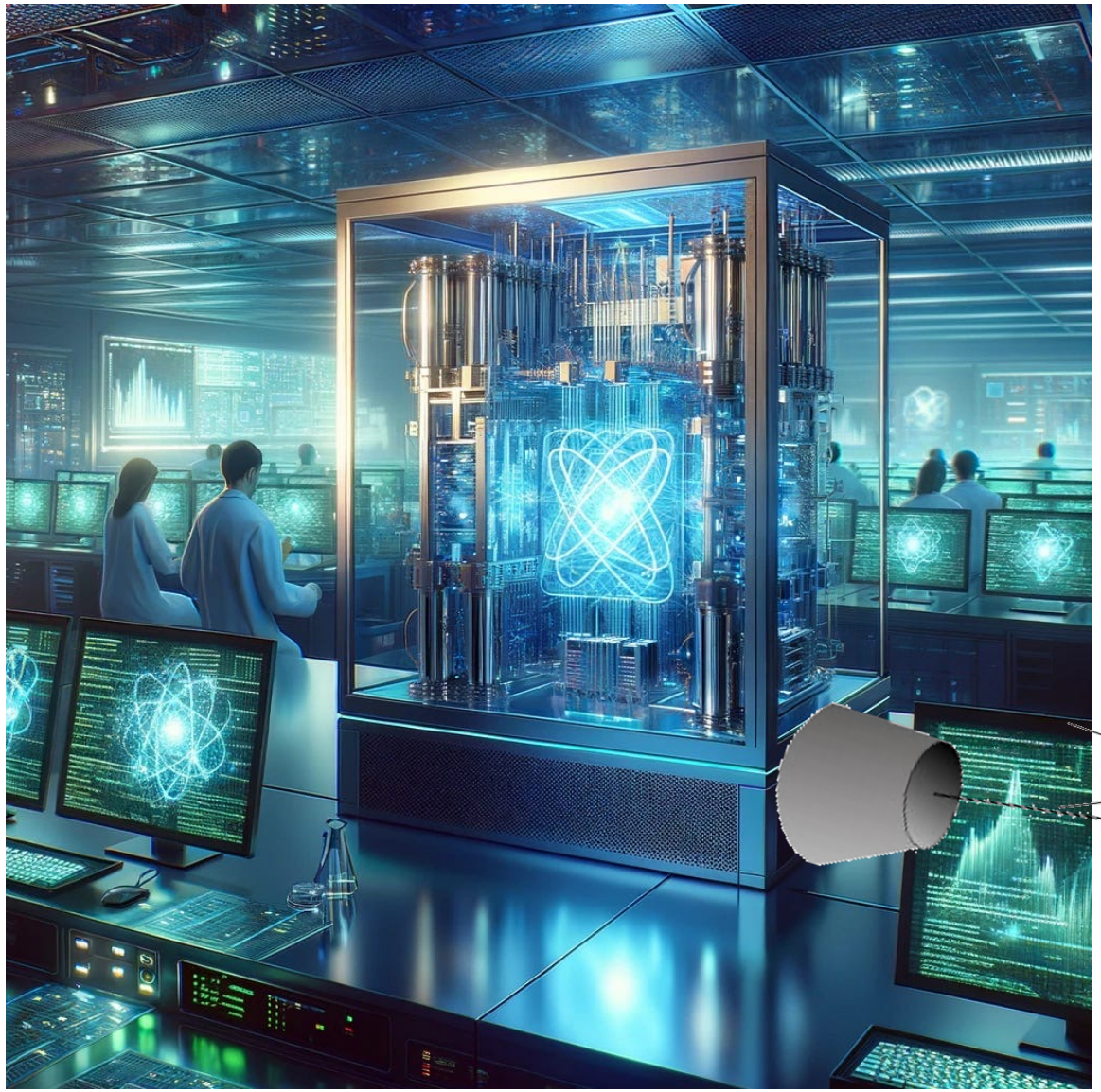
Ravel

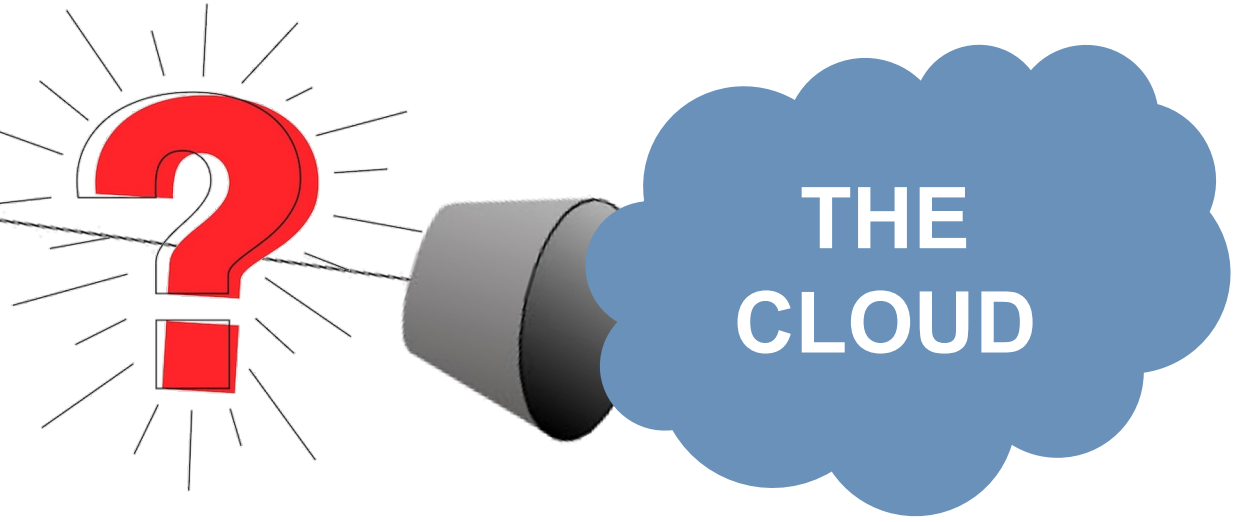


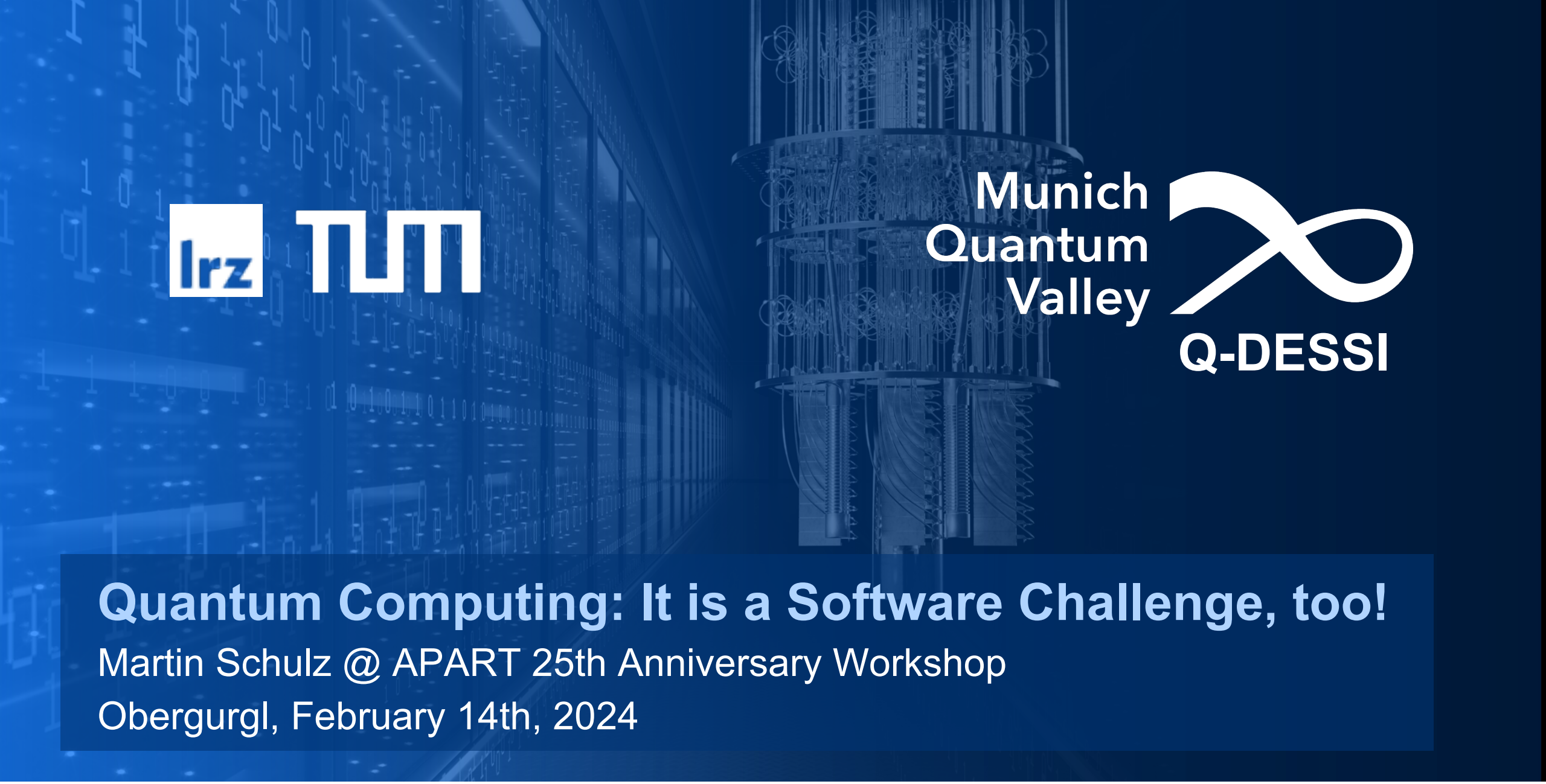
And Now to Something Completely Different!

And Now to Something Completely Different!

Lot's of Excitement Around Quantum Computing







Quantum Computing: It is a Software Challenge, too!

Martin Schulz @ APART 25th Anniversary Workshop

Obergurgl, February 14th, 2024

The Promise of Quantum Computing

System working on quantum mechanical principles

- Superposition = Multiple States at one time (until measured)
- Entanglement = Correlation between states in a system
- Probabilistic

Basic unit = 1 Qubit

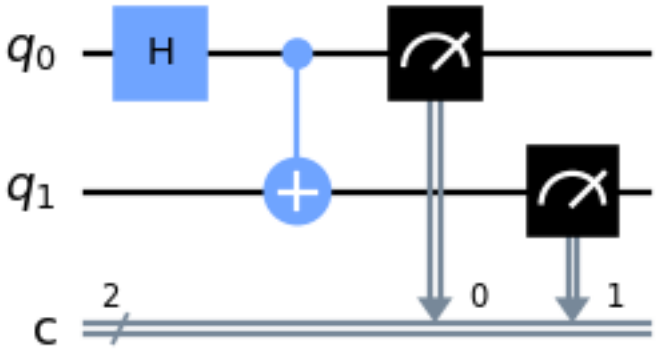
- Can be in superposed and/or entangled state
- When read: 0 or 1 only (rest collapses)

Programming with gates

- Operations on qubits
- Different systems have different gates

Quantum Advantage

- Some quantum algorithms require exponentially less steps than classical algorithms



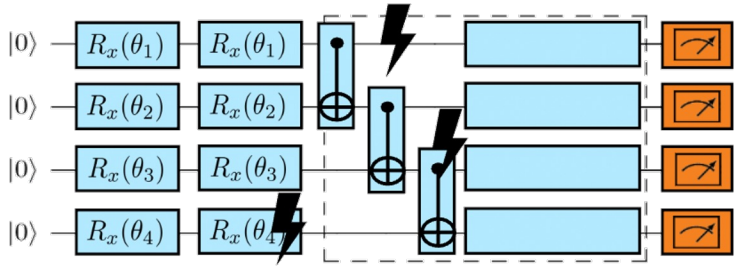
From: https://docs.quantum.ibm.com/api/qiskit/qiskit.circuit.QuantumCircuit

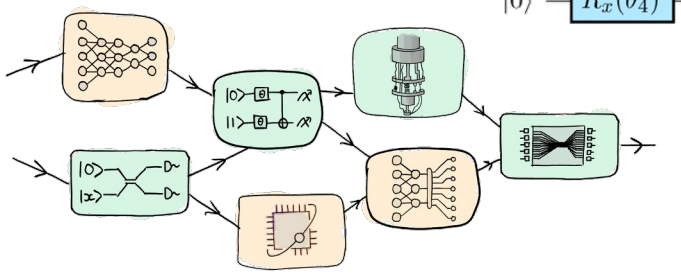
Applications for Quantum Computing

Main Application Domains

- Quantum simulation of quantum systems
- Quantum optimization
- Quantum machine learning
- Quantum linear systems

$$\mathcal{N}(\rho) = \sum_{i=0}^{n} K_i \rho K_i^\dagger$$

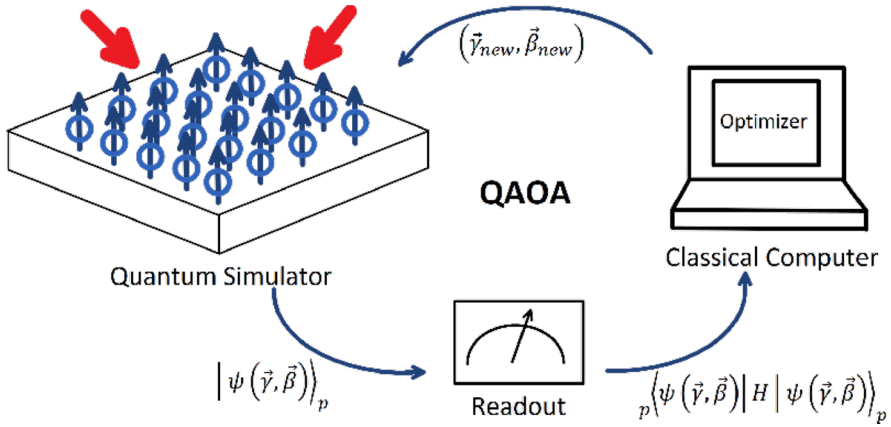




PENNYLANE

Problems:

- Small number of qubits
- Noisy systems
- Only few working algorithms
- Specialized programming
- Still treated as physics experiments



The Munich Quantum Valley initiative develops quantum computation and quantum technologies in Bavaria.

Superconducting. Ion. Neutral Atom. Quantum-HPC.

BAdW Bayerische Akademie der Wissenschaften · DLR · FAU Friedrich-Alexander-Universität Erlangen-Nürnberg · Fraunhofer · LMU Ludwig-Maximilians-Universität München · Max Planck Gesellschaft · Technische Universität München

Mission of MQV

Architect, Design, Implement and Deploy Full Stack Quantum Computing Systems In and for Bavaria

BAdW Bayerische Akademie der Wissenschaften | DLR | FAU Friedrich-Alexander-Universität Erlangen-Nürnberg | Fraunhofer | LMU Ludwig-Maximilians-Universität München | Max Planck Gesellschaft | Technische Universität München

Mission of Q-DESSI

Develop the Central Open Source Software Stack for Full Stack Quantum Computing Systems In and for Bavaria

For the MQV project

For and co-funded by DAQC, Q-Exa, QuaST, MUNIQC-ATOMS/SC, …

For the for QC community

BAdW Bayerische Akademie der Wissenschaften · DLR · FAU Friedrich-Alexander-Universität Erlangen-Nürnberg · Fraunhofer · LMU Ludwig-Maximilians-Universität München · Max Planck Gesellschaft · Technische Universität München

Mission and Goals of Q-DESSI

Munich Quantum Valley

Q-DESSI provides programming & runtime env.

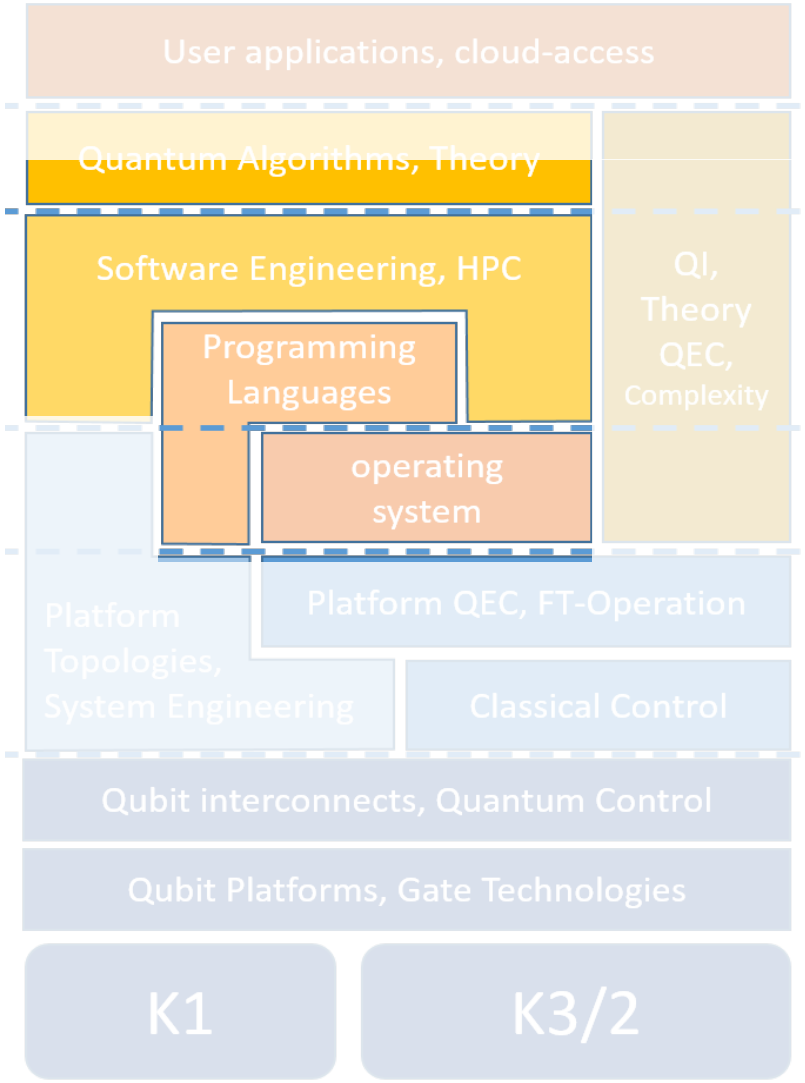
- Compilers, Optimizers and Tools
- Secure multi-user environments
- Internal and external resource management

Q-DESSI provides integration with existing systems

- Host platforms for control and analysis
- QC as accelerator for HPC
- Onloading of optimization/error correction to HPC

Four Focus Areas (aka. Central Tasks)

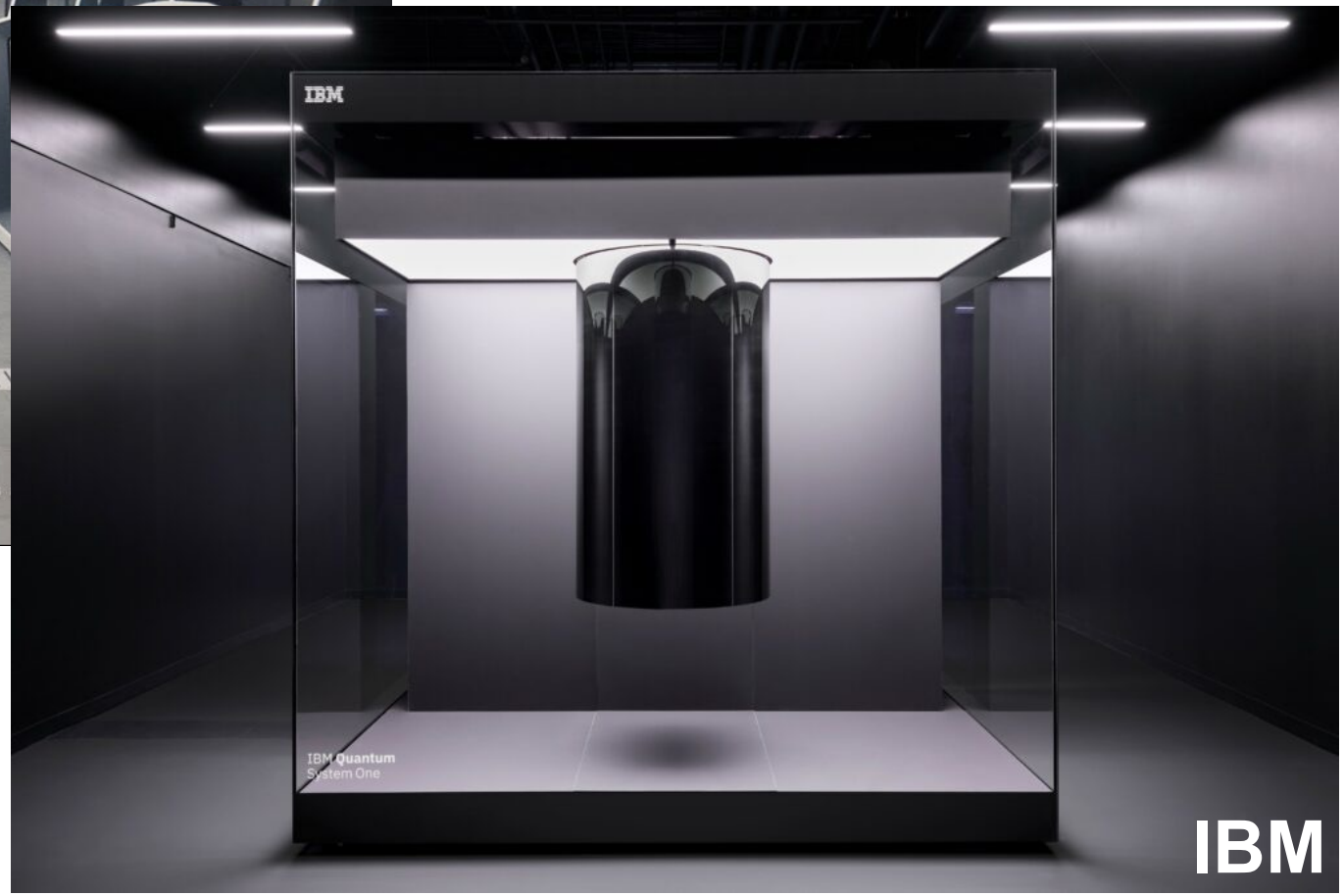
- Programming Environments, Compilers and Tools
- Quantum OS and Runtime
- Quantum Control Microarchitecture & Firmware
- Integration with HPC & Cloud Systems



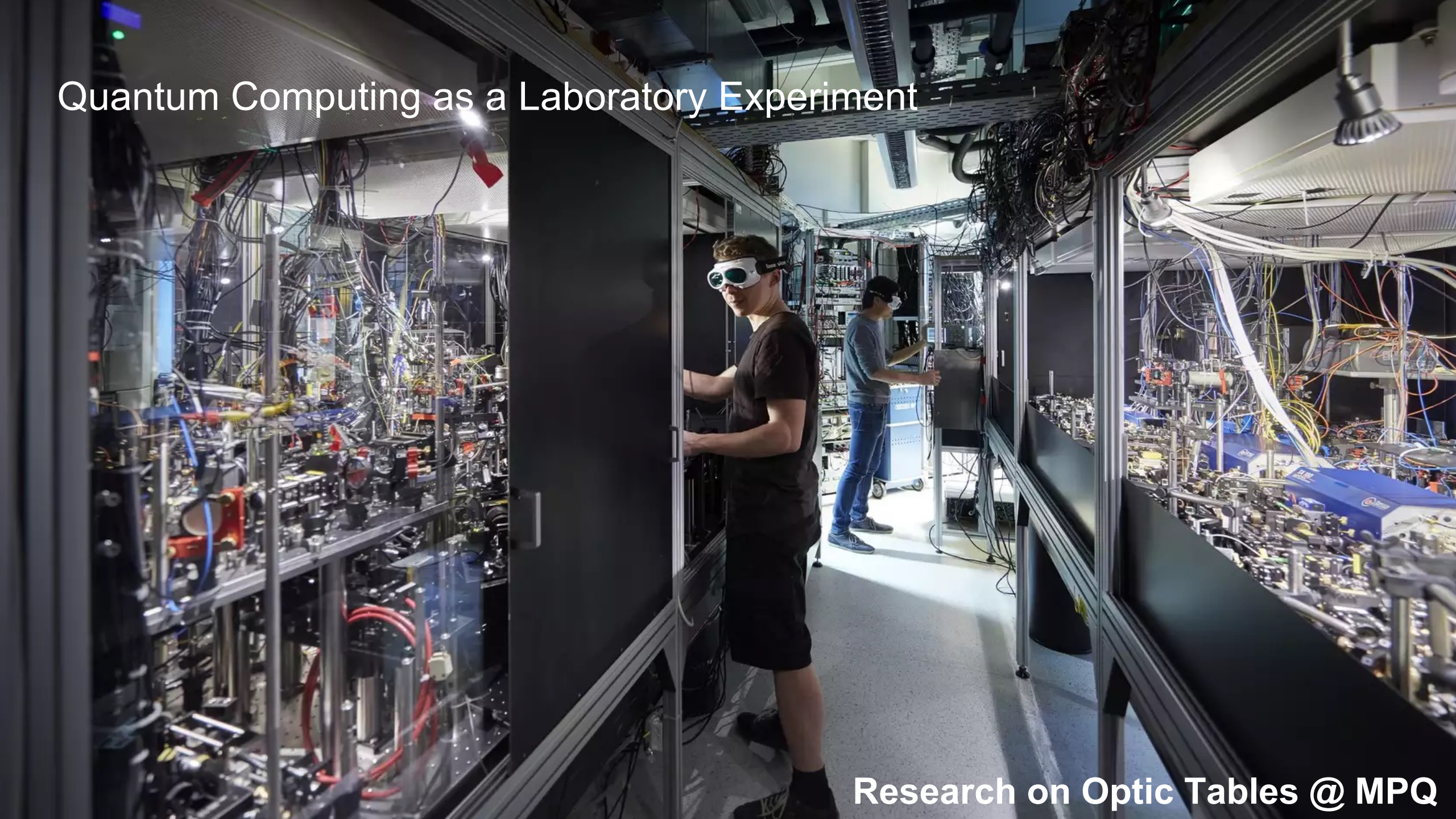
So, Where Do We Stand?

How will they look like?

Artistics Renderings of Quantum Computers



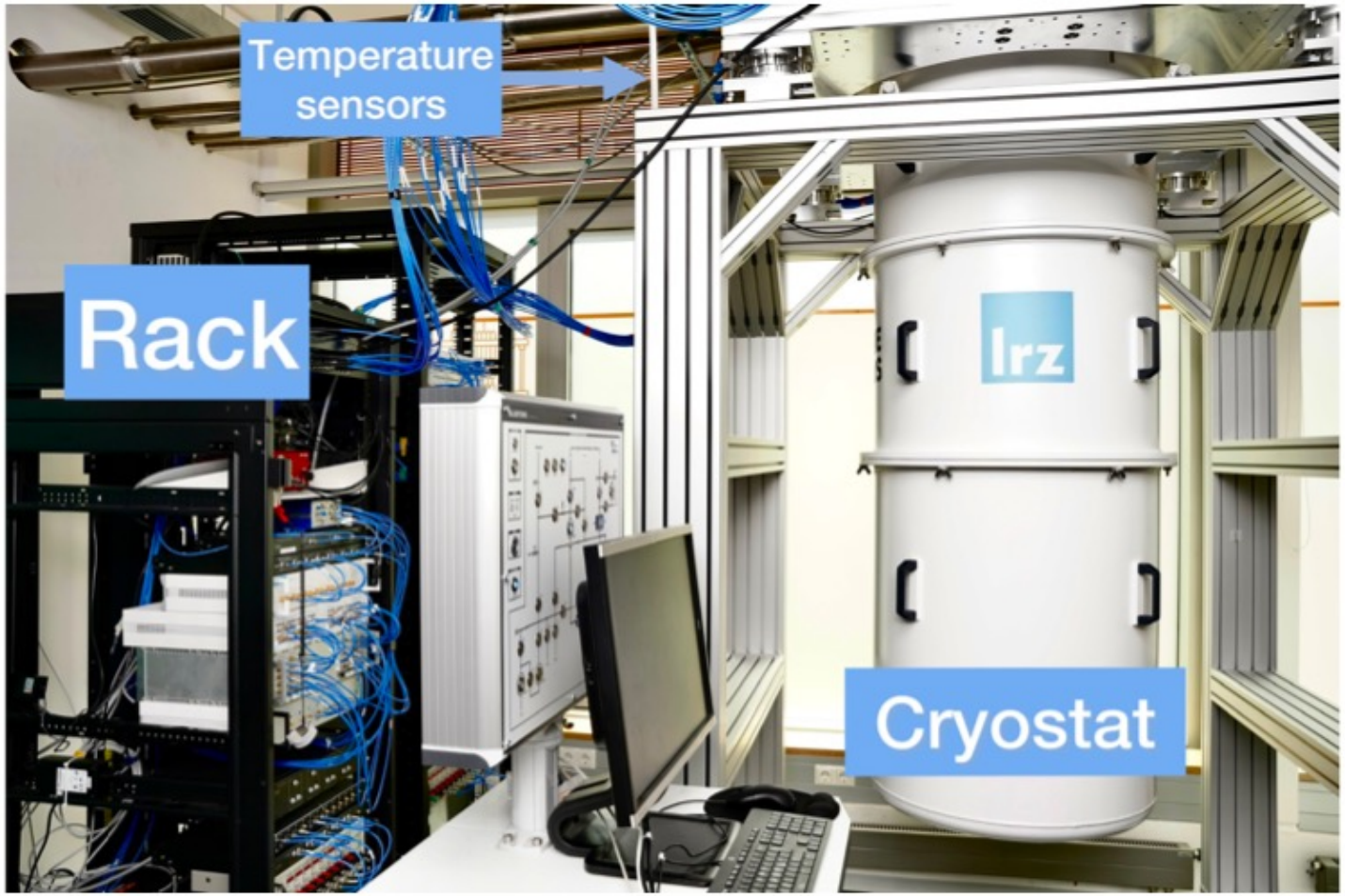
Quantum Computing as a Laboratory Experiment



Research on Optic Tables @ MPQ

First ”Commercial” Systems at a Compute Center

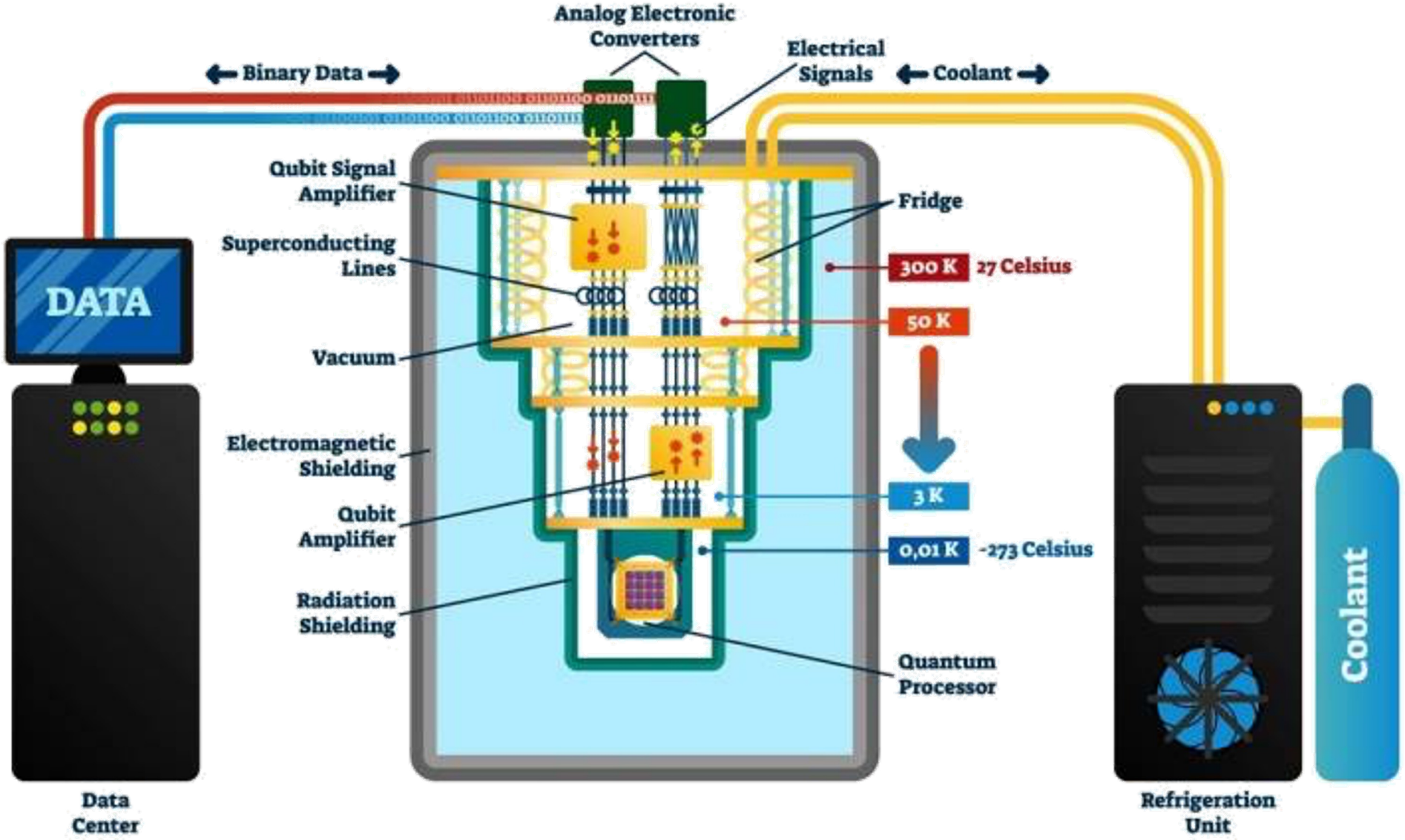
Examples in the LRZ Quantum Integration Centre (QIC)



IQM @ LRZ

AQT @ LRZ

Cryo-Setup for a Superconducting System

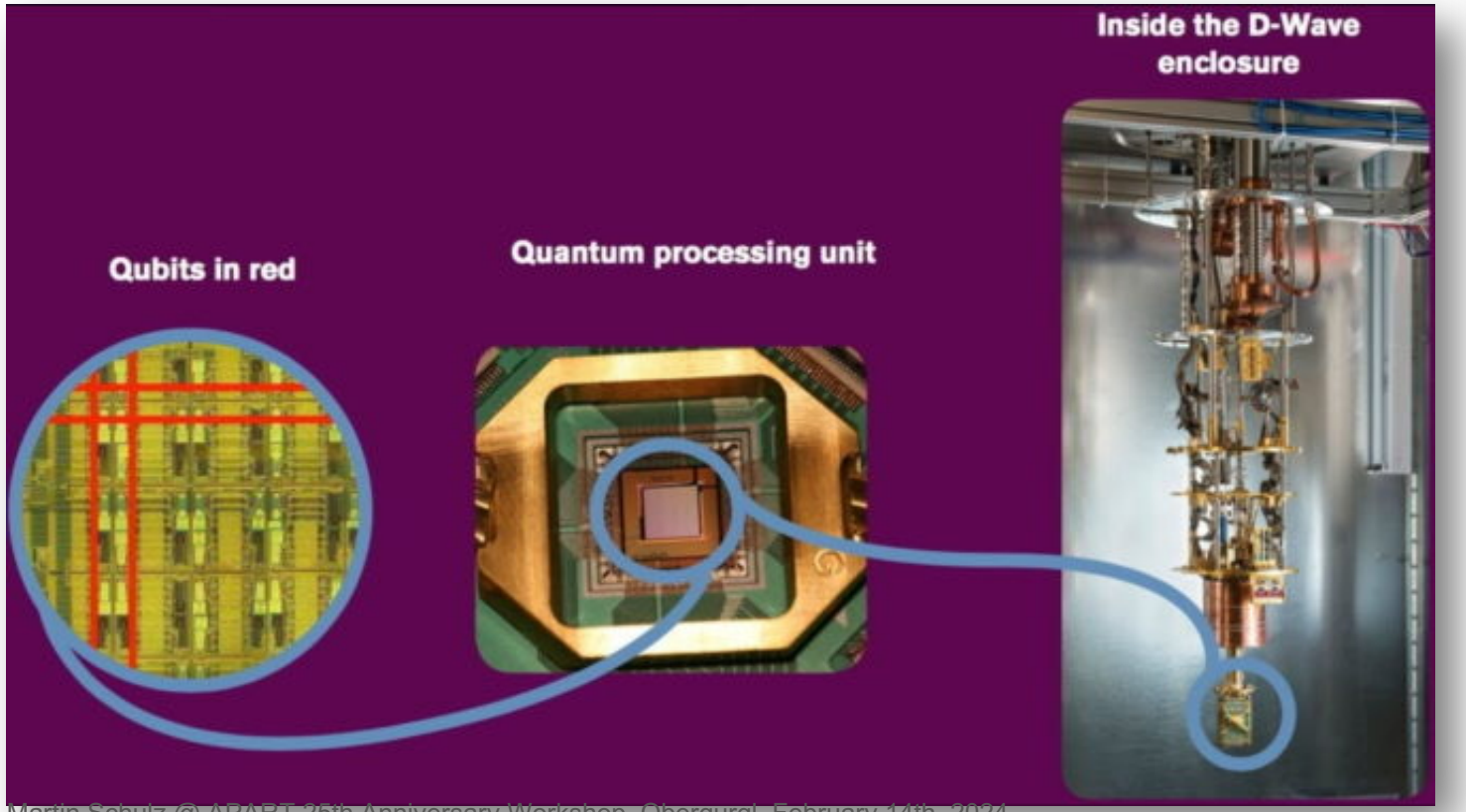


https://www.thebroadcastbridge.com/content/entry/14159/instant-answers-from-the-universe

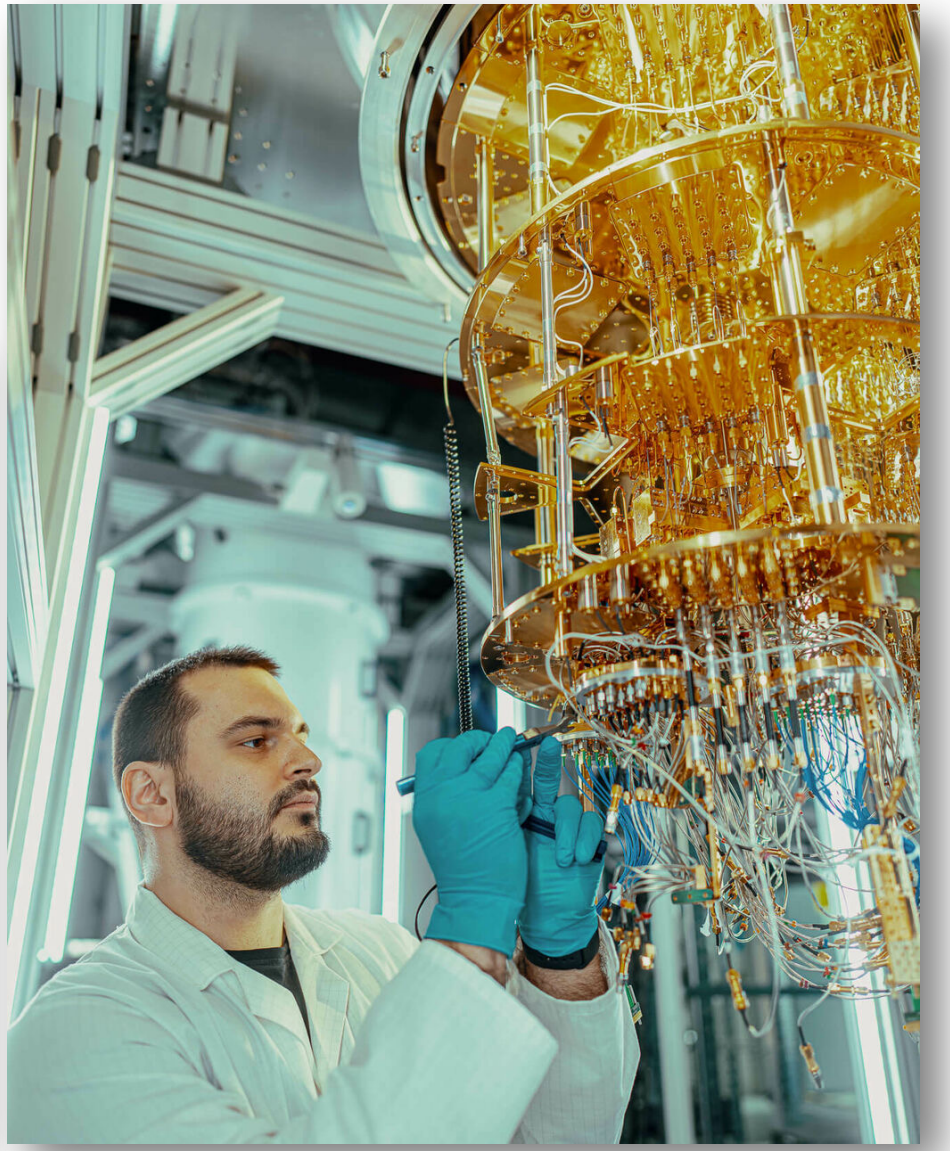
How does it work?

The Quantum Processing Unit

- Qubits are manufactured onto a chip
- Directly controlled from the outside
  - No internal control architecture
- Noisy intermediate scale quantum (NISQ)

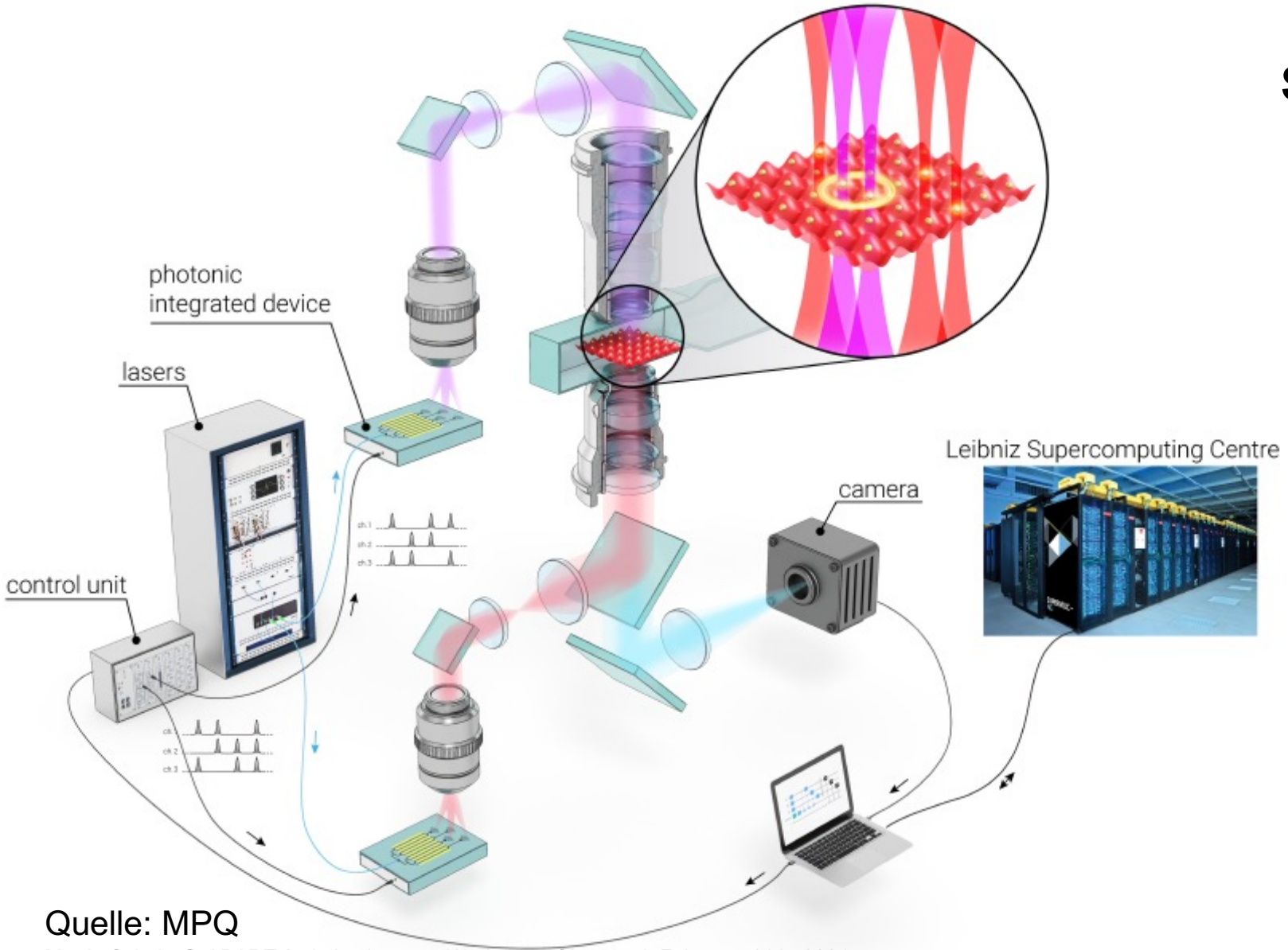


https://blogs.nvidia.com/blog/what-is-a-qpu/



Source: IQM

Second Example: Neutral Atoms



Quelle: MPQ

Similar Structure

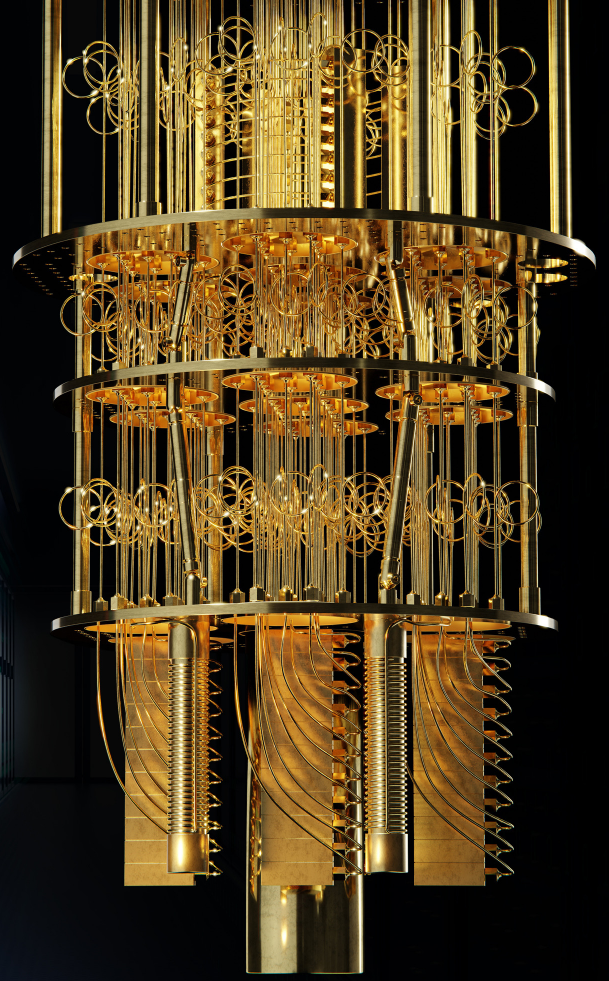
- Different Control
  - Laser instead of Microwave
- Different Qubits
  - Individual Atoms

- Again: separate external control system
- Again: classic compute to control the system

Quantum computers will not replace HPC.

Quantum computers will not replace HPC.

Quantum *accelerators* are HPC.



Why HPC-QC?

Quantum
Computing
=
High-Performance
Computing

New compute capability that adds to
the supercomputing portfolio.

Strategy: Quantum Computing as Accelerator for HPC

Quantum Computing as a stand-alone system not viable at growing scales

- Complex control systems
- Data staging and post-processing
- Targeted towards very specific workloads and kernels
- Tight interactions needed for variational algorithms
- Complex compilation and runtime environment with high demands need HPC

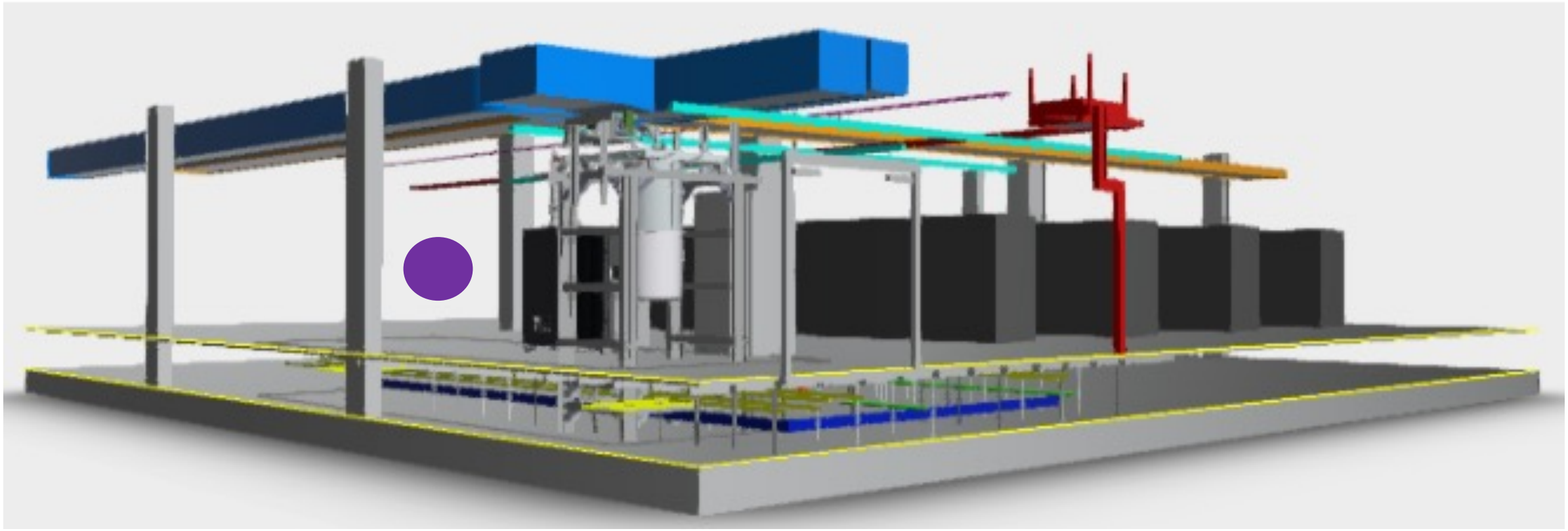
Usage exclusively as accelerator for HPC workloads

- Intended for fine-grained kernels within larger applications or workloads
- Similar to other accelerators, on-node (like GPUs, FPGAs) or dissaggregated (like AI HW)

Consequences: HPC and QC as a single HPCQC system

- Requires tools and models to extract application components relevant for acceleration
- Requires easy access for HPC community to QC programming (or library usage)
- Requires integration into a single programming environment
- Requires close hardware integration (single system) for latencies and management
- Requires unified user access/management/experience

Location of Production QC systems in LRZ Compute Cube



Planned location of System Q-Exa (shown) and Euro-Q-Exa system (to the left of Q-Exa, purple dot).

Quantum Systems at LRZ

Superconducting

System 1: 5 qubits (R&D system)
Vendor: IQM

System 2: ca. 20 qubits
Vendor: IQM
Accessible: H1-2024 to select research partners

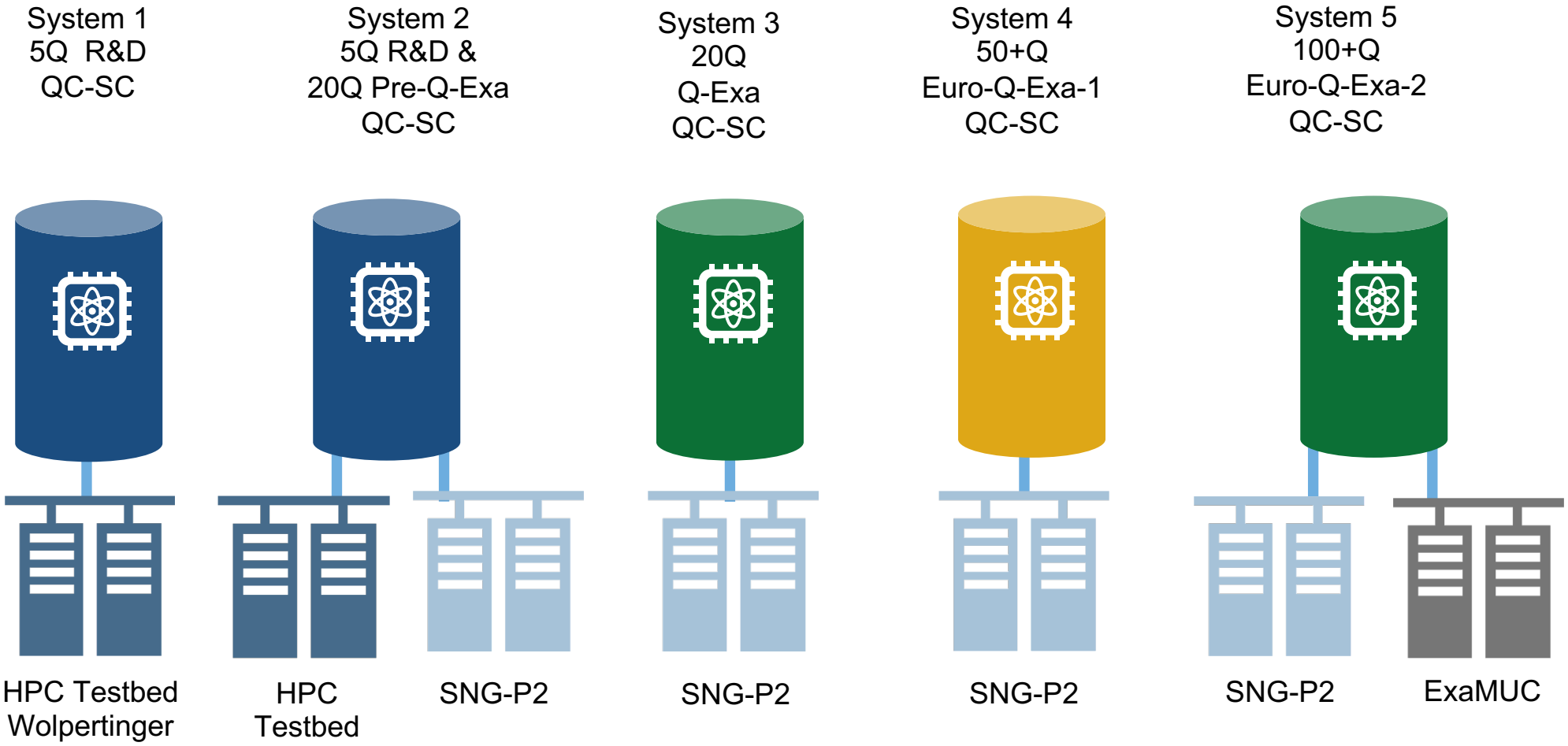
System 3: 20 qubits
Vendor: IQM
Accessible: Late H1-2024 to Bavarian, German and European users in pilot phase initially, wider access in 2024

System 4: 50+ qubits
Vendor: TBD
Accessible: Early H2-2025 to Bavarian, German and European users

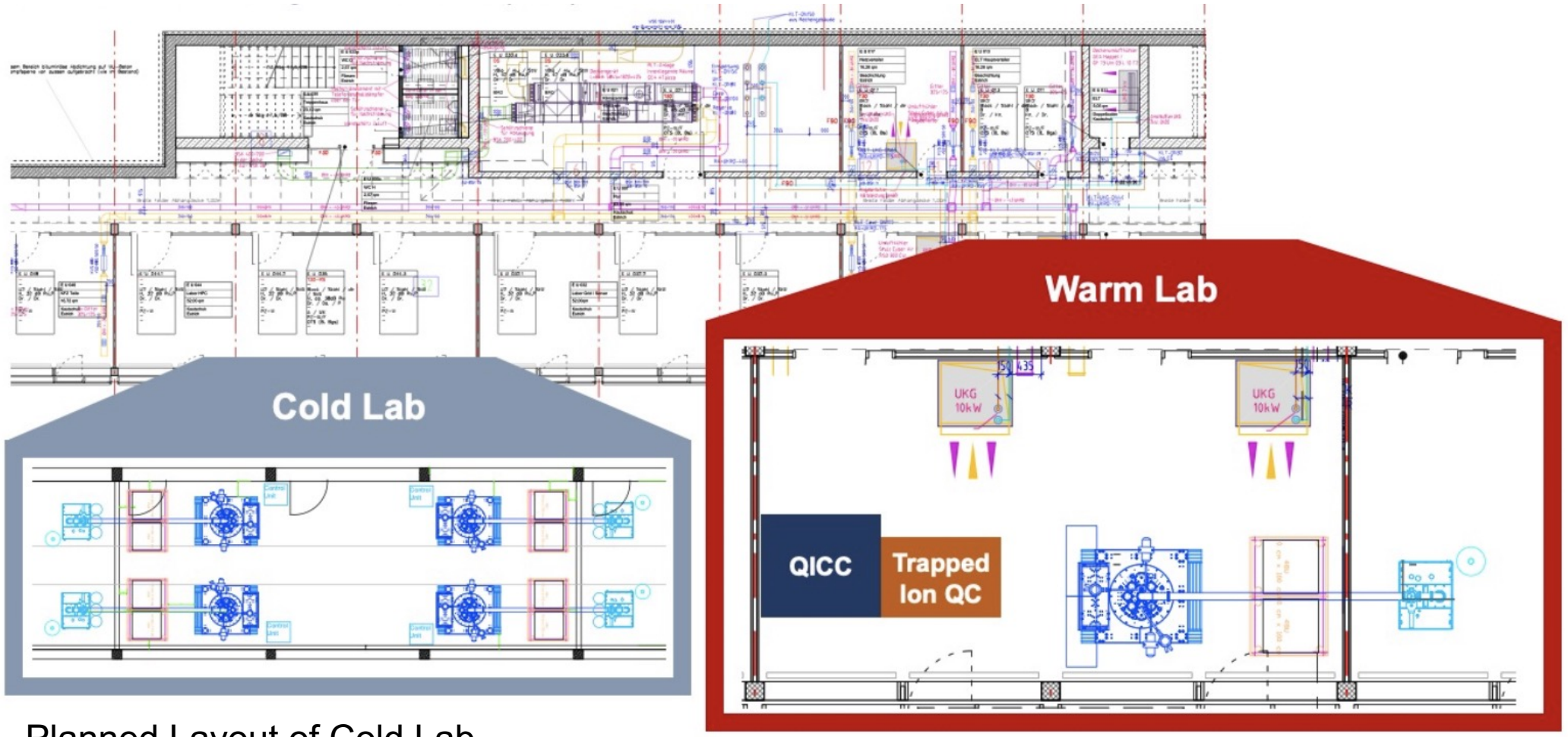
System 5: 100+ qubits
Vendor: TBD
Accessible: Early H1-2026 to Bavarian, German and European users

LRZ Quantum Systems

Superconducting Systems Roadmap with HPC



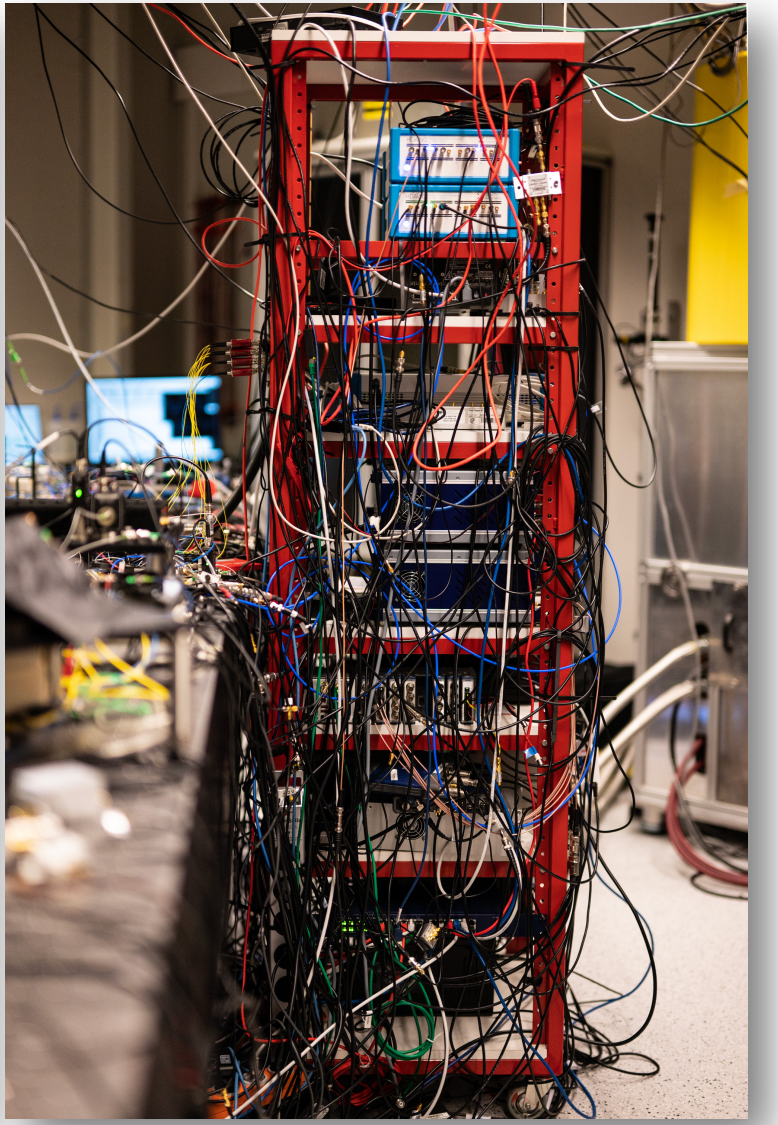
Quantum Integration Centre (QIC)



Planned Layout of Cold Lab

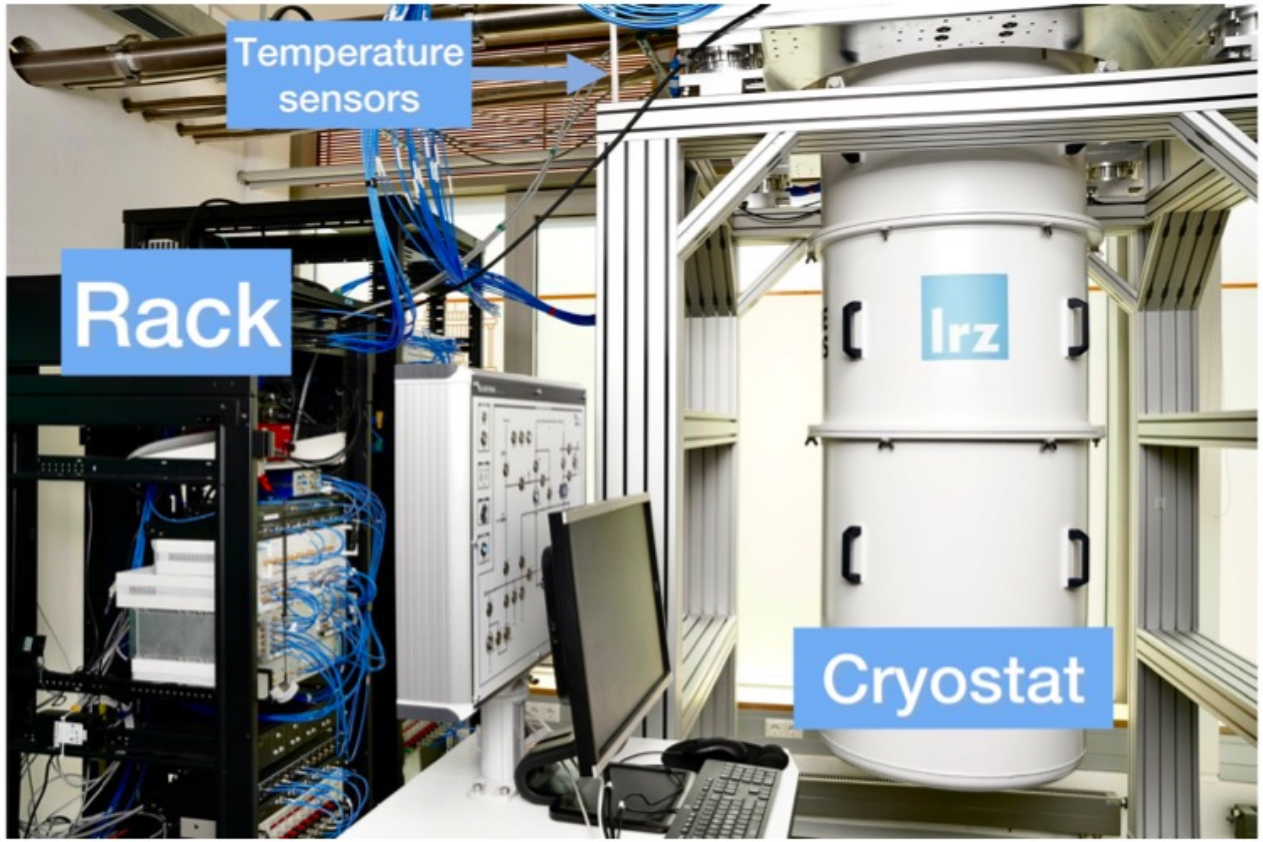
Lab Setup

Quantum Integration Centre (QIC)



Source: MPQ

Integration into HPC Center Telemetry



HPC Center Monitoring

System and Infrastructure Health
Application Performance

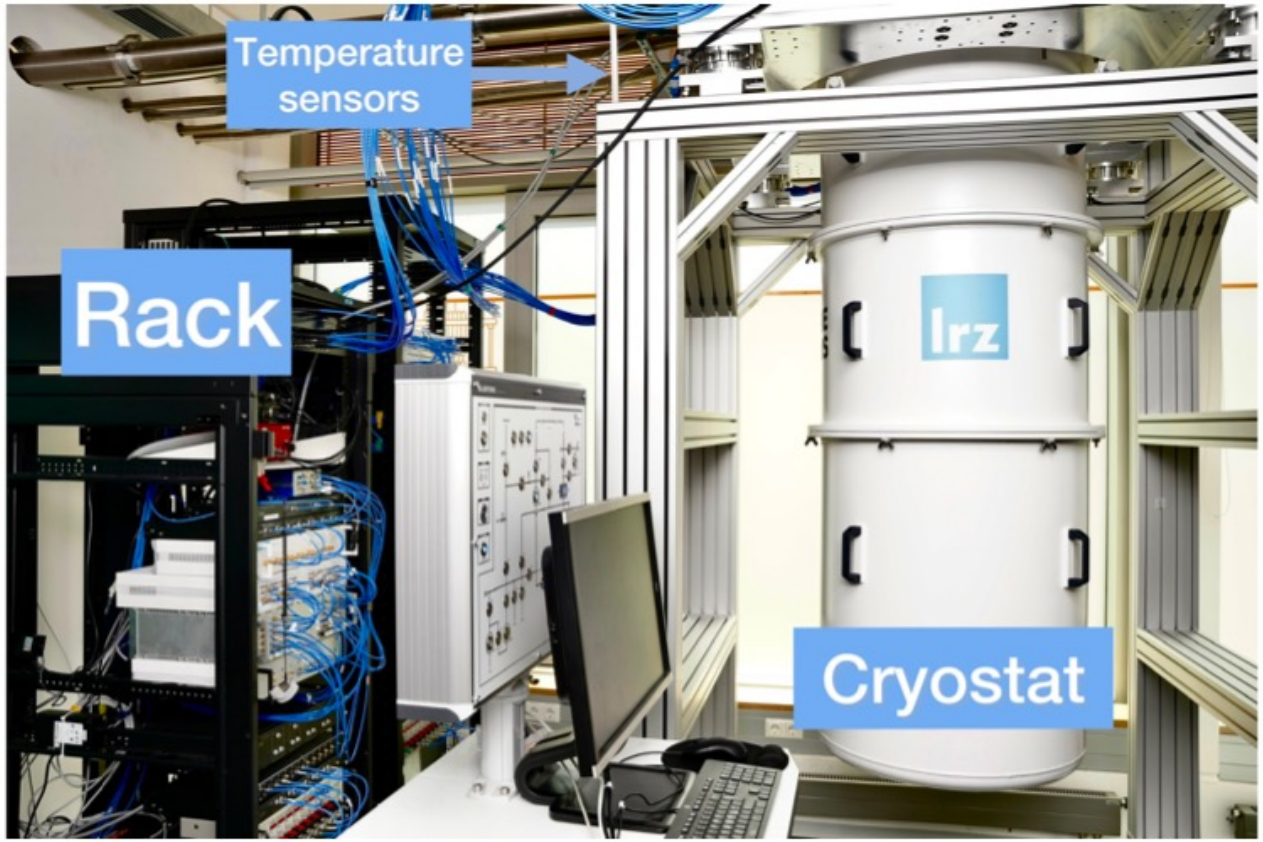
Inclusion of QC

From Cryostat to Control Electronics
IoT Infrastructure needed
Plus: Key metrics like fidelity

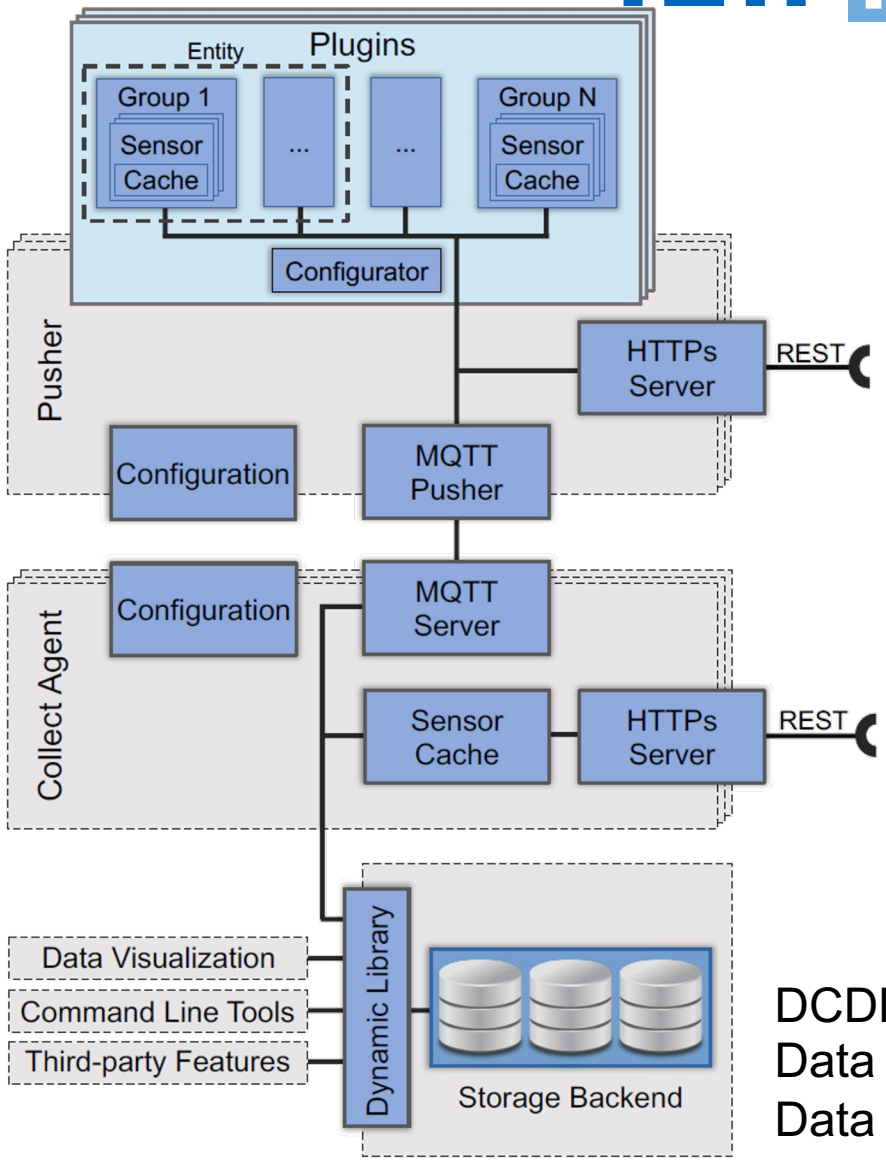
Operational Data Analytics

Towards predictive / hands-off maintenance

Integration into HPC Center Telemetry



For more on this: talk by Hossam Ahmed at the WIHPQC23 workshop today



DCDB
Data Center
Data Base

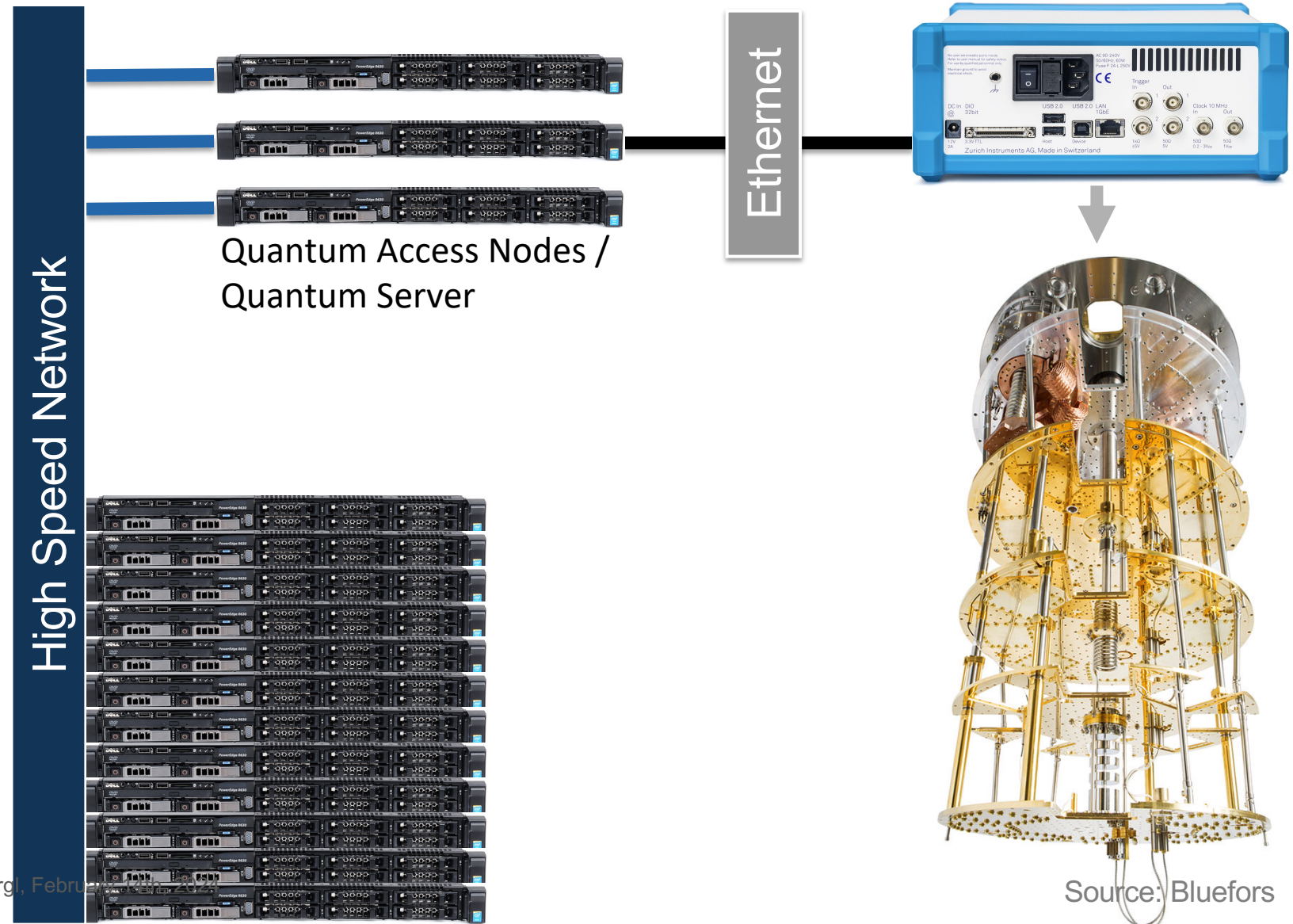
Integration von QC und HPC

Connection to the HPC network

Direct connection to QC controller

BUT:

It has to be a full stack solution!



Vision for HPCQC

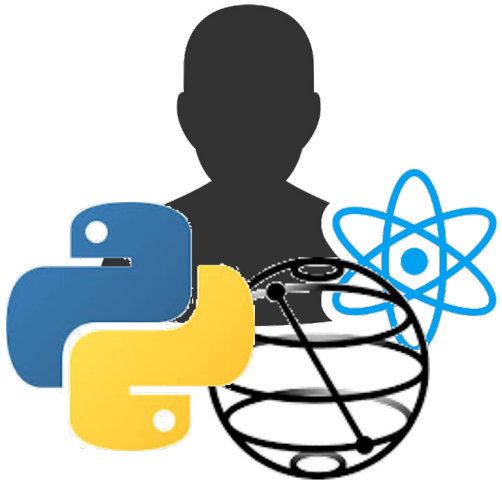
Infrastructure	Standalone system with own monitoring	→	Integrated in data center with cooling, power/energy, monitoring
Hardware	Quantum Computing Appliance + HPC System	→	Single system with integrated QC accelerator(s)
Software	Two separate software stacks with “glue scripts”	→	Single-source programming with high-level abstractions
Applications	Algorithms with quadratic speedup and separate QC and HPC applications	→	Algos with faster speedup and truly hybrid applications
Culture	Computer Science, Math, Electrical Engineering, Physics communities	→	Hybrid HPCQC community

Some HPCQC End Goals

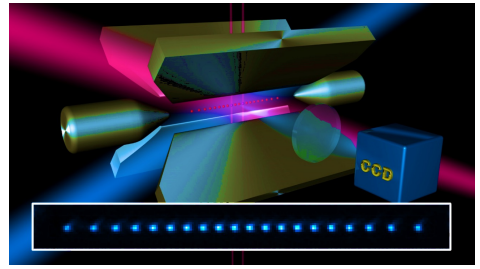
Infrastructure	Standalone system with own monitoring	→	Integrated in data center with cooling, power/energy, monitoring
Hardware	Quantum Computing Appliance + HPC System	→	Single system with integrated QC accelerator(s)
Software	Two separate software stacks with “glue scripts”	→	Single-source programming with high-level abstractions
Applications	Algorithms with quadratic speedup and separate QC and HPC applications	→	Algos with faster speedup and truly hybrid applications
Culture	Computer Science, Math, Electrical Engineering, Physics communities	→	Hybrid HPCQC community

Current State of Software

Individual Physics Researcher



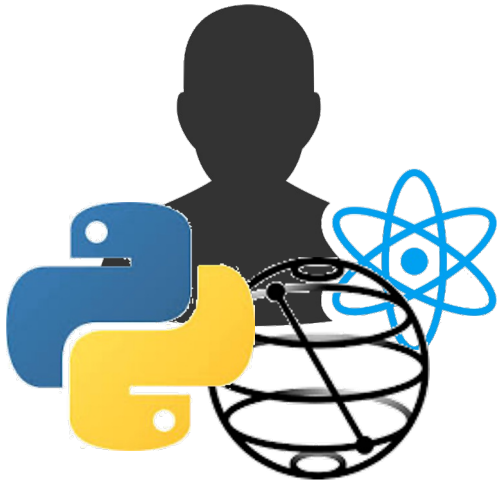
Quantum Device



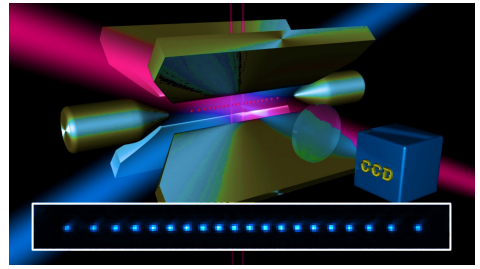
Supporting the Needed Transition the Laboratory to Production

Challenges in Current Software Setup

Individual Physics Researcher



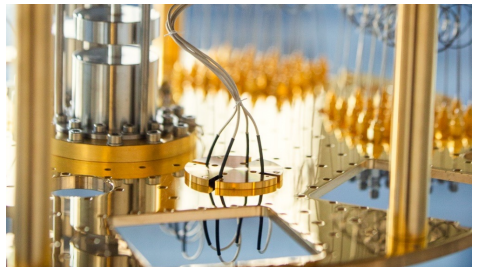
Quantum Device

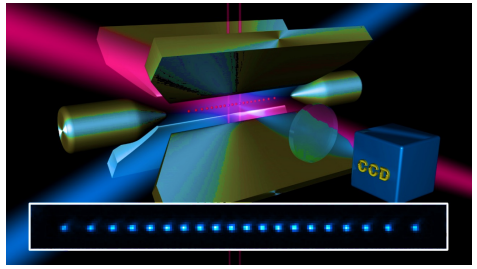


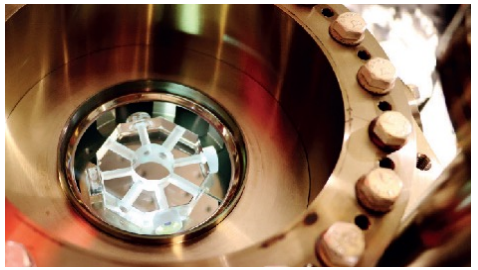
- Need Discipline Users
- Need Faster Toolkits
- Need More Comp. Capacity
- Need Standard Interfaces
- Need Near QC Compute
- Need Multiple Backends

Requirements

Wide User Communities





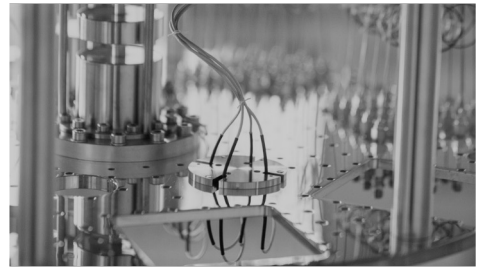


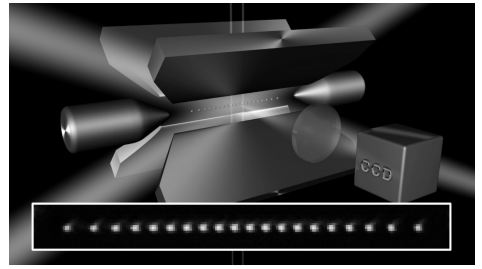
Many/Different Quantum Devices

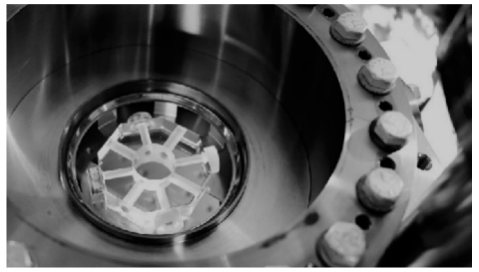
Requirements

Wide User Communities

Comprehensive Software Stack



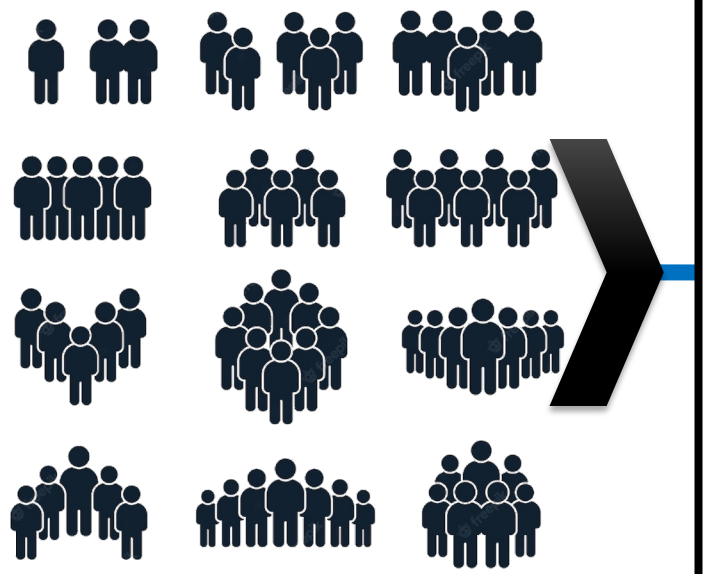




Many/Different Quantum Devices

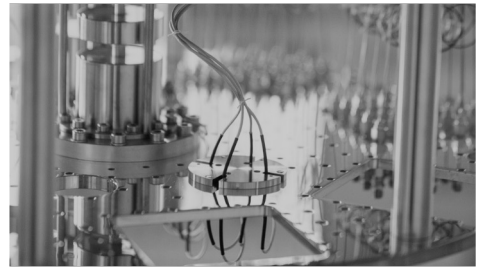
Requirements

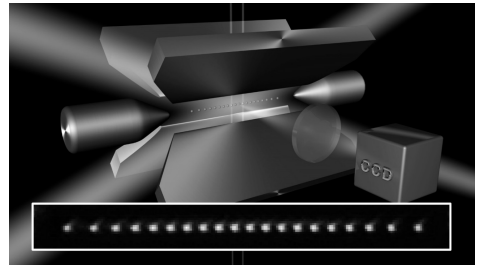
Wide (HPC) User Communities

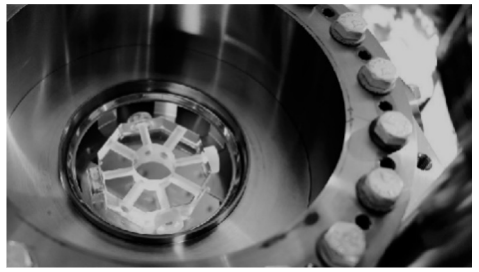


Comprehensive Software Stack

integrated into HPC Environments



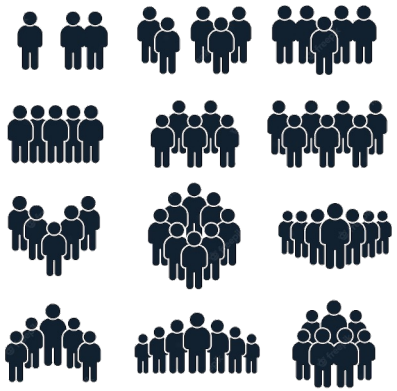




Many/Different Quantum Devices

MQSS Overview

Wide HPC User Communities



Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

HPC System With Quantum Accelerator

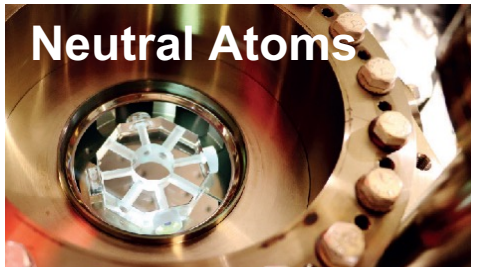
Scheduling Resource Management

Quantum Compiler based on QIR

Comprehensive Toolkits, Optimizers, Verifiers, Simulators

Figures of Merit and Constraints

QDMI = Quantum Backened Interface

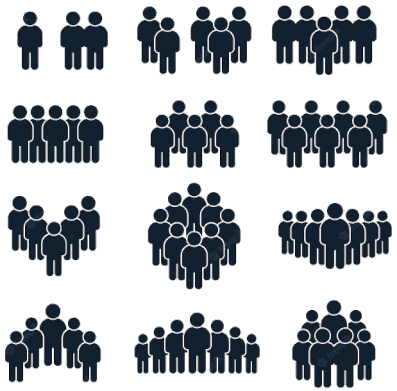


Enabling Domain User Communities to Compute on Quantum Devices

QCs & Simulators

Front-End / Languages

Wide HPC User Communities



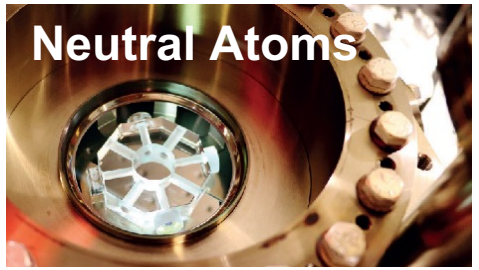
Hybrid/Quantum Programming Toolkits / Libraries

Different Programming Models

- Standard approaches like Qiskit/Pennylane
- Higher-level abstractions
  - Incl. those provided by QLM
- New developments
- HPCQC approaches, like OpenMP
- Domain specific support

Decoupled from compilation stack

- Multiple front-ends, one backend
- Translation into standard format
  - QIR establishing itself as standard
- Result: Templated circuits



Enabling Domain User Communities to Compute on Quantum Devices

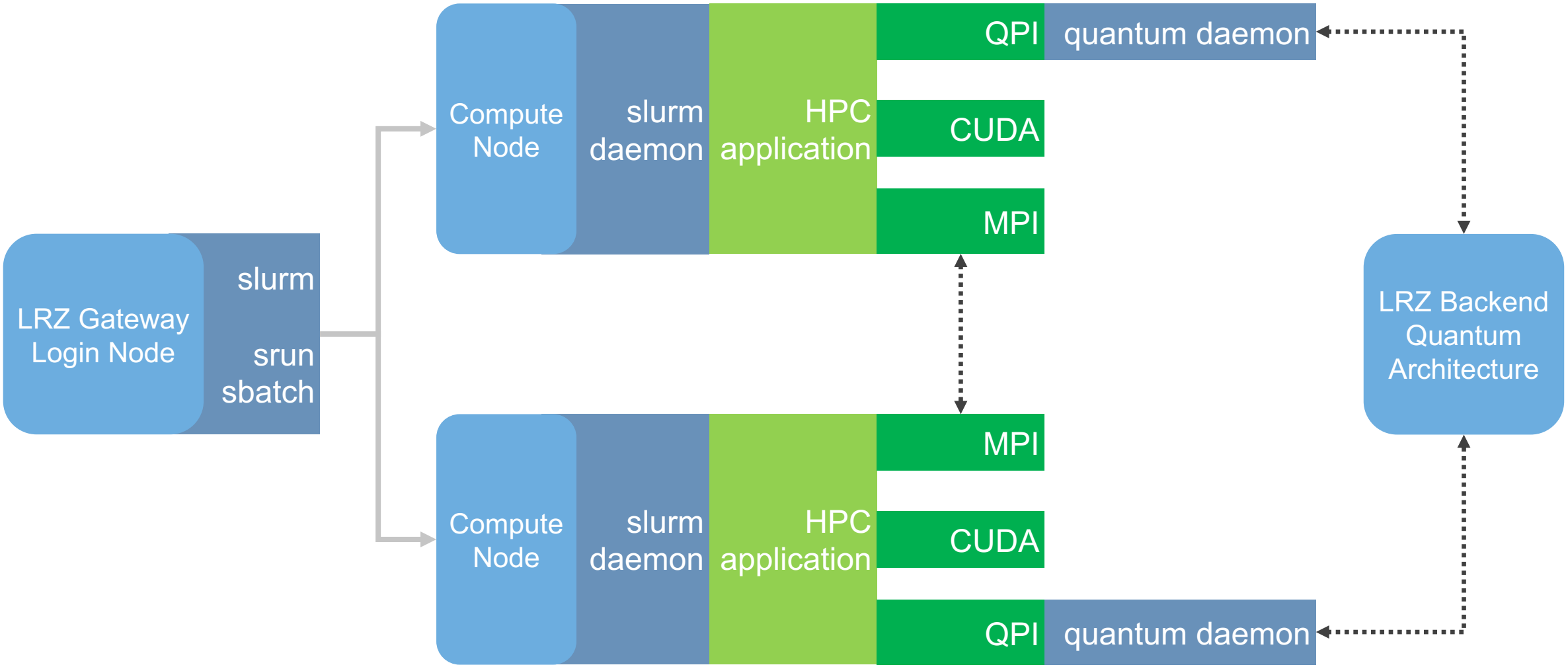
QCs & Simulators

Quantum Programming Interface

- Aims to provide similar abstraction to Qiskit
- Abstracts architecture and vendors to avoid lock in
- Users are legacy HPC applications
- State machine interface for quantum task programming
- Maps well to task-offload model and doesn't force data structure

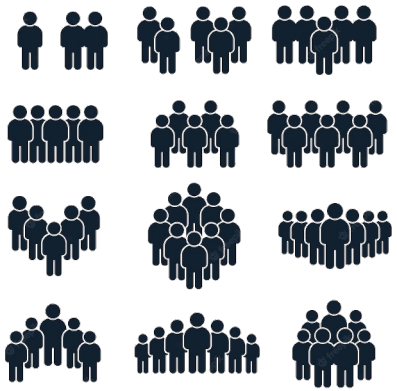
```c
#define QPI_1
#include <qpi.h>
#include <stdio.h>


void bell_0() {
    Qcircuit circuit;
    Qstatus status;

    int states = 4;
    int shots = 1000;

    // 4 states can exist with 2 qubits
    int output[states];

    qCircuitBegin(&circuit);

    qH(0);
    qCX(0, 1);

    qMeasure_all();

    qCircuitEnd();

    qExecute(circuit, shots, &status);
    qWait(status);

    qRead(status, QPI_READ_ALL_STATES, (int*)&output);

    for(int state_idx=0; state_idx < states; state_idx++) {
        printf("|%d>: %d", state_idx, output[state_idx]);
    }
}
```

HPC Accessing QC via QPI



Portal Access (aka. “Cloud Access”)

Wide HPC User Communities

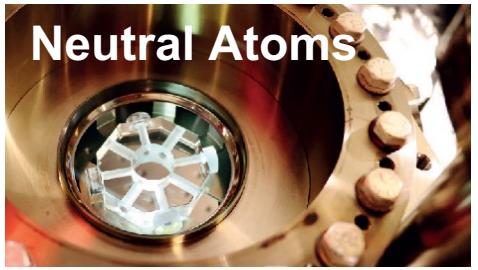


Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

Portal for individual experiments

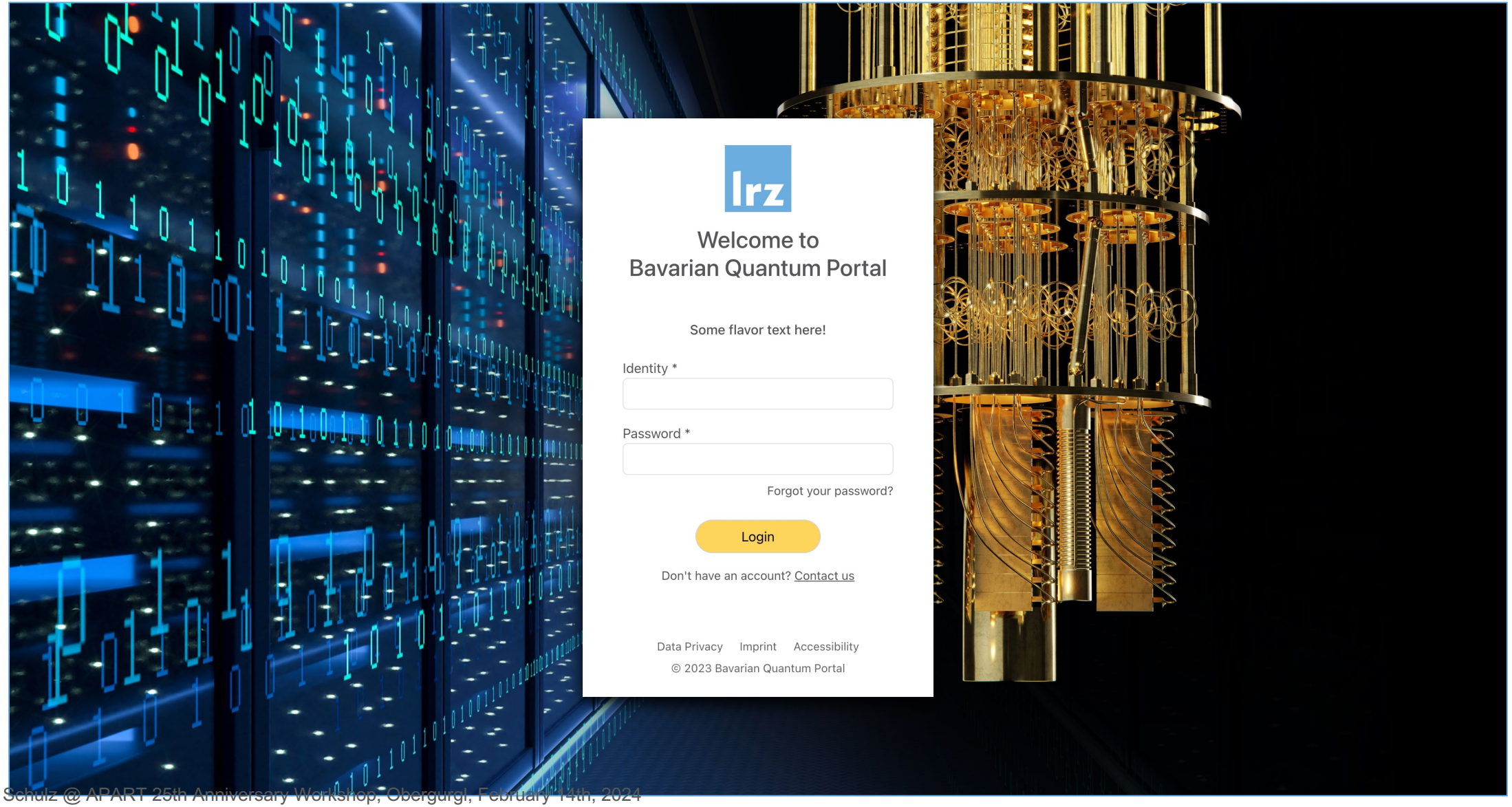
- Authentication and authorization
- Web-based access
- Simple, but high latency



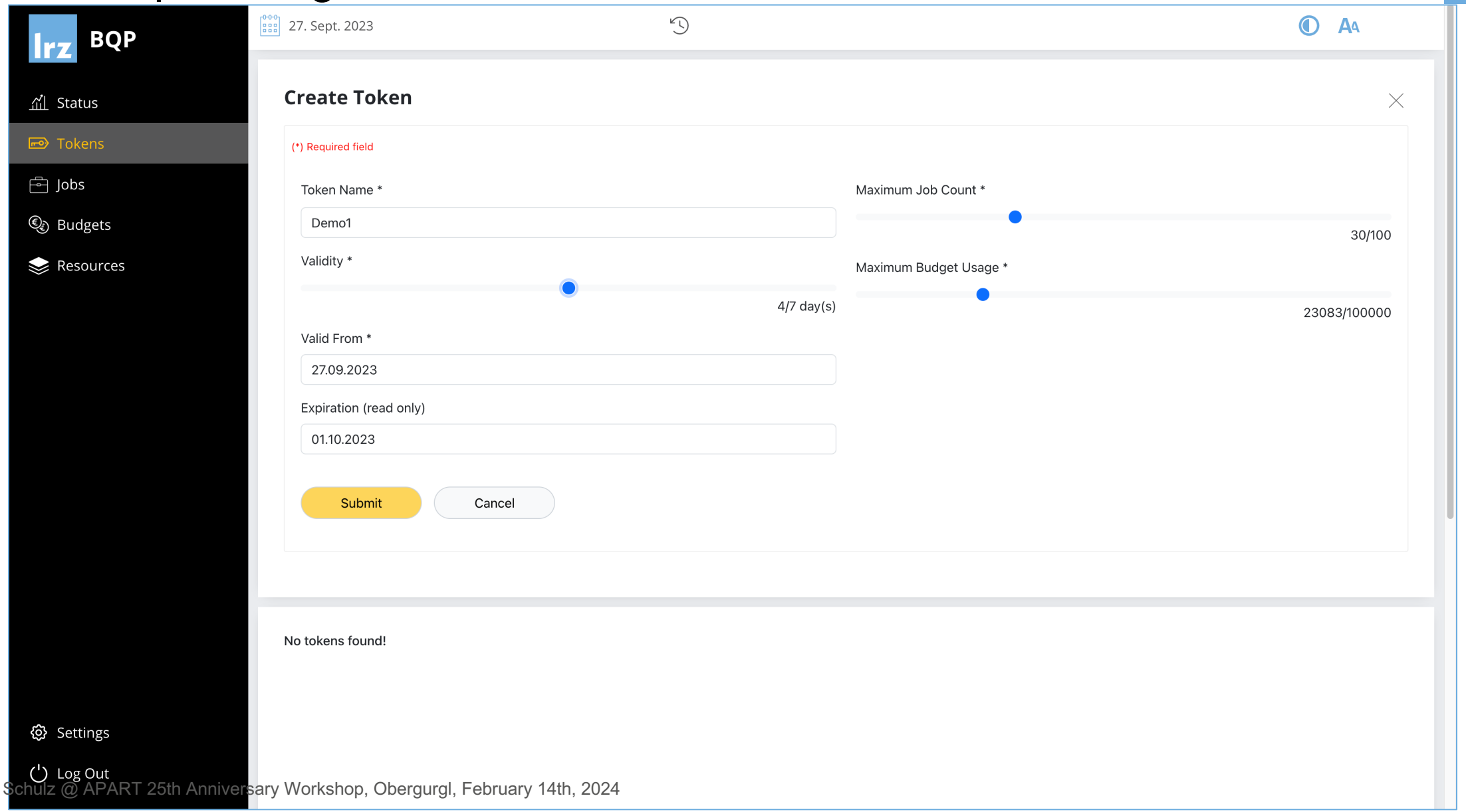
Enabling Domain User Communities to Compute on Quantum Devices

QCs & Simulators

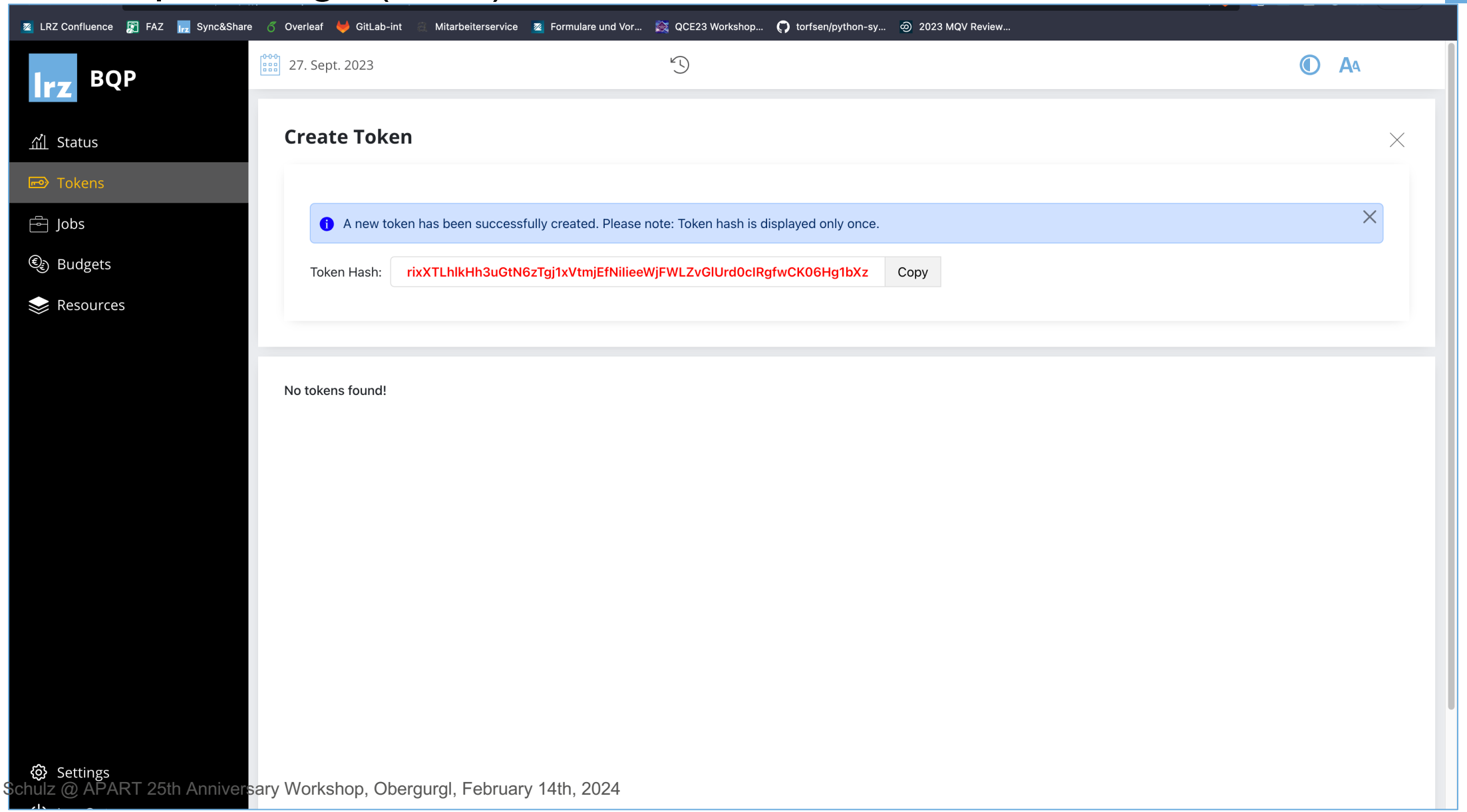
The Bavarian Quantum Portal (BQP)



BQP Sample Usage



BQP Sample Usage (cont.)



BQP Qiskit Provider

```python
from bavarian_quantum_portal_provider import BQPProvider

from pprint import pprint

import termplotlib as tpl


if __name__ == "__main__":
    c = QuantumCircuit(5, 5)

    c.h(2)
    c.cx(2, 1)
    c.cx(2, 0)
    c.cx(2, 3)
    c.cx(2, 4)

    c.measure_all()

    print(c.draw())

    with open("token.txt") as tokenfile:
        provider = BQPProvider(token=tokenfile.read()[:-1])

    backend = provider.get_backend("Test Q5")

    job = backend.run(c, shots=1000)

    result = job.result()

    pprint(result.get_counts())

    proper = {
        k: v for k, v in sorted(result.get_counts().items(), key=lambda item: item[0])
    }

    fig = tpl.figure()
    fig.hist(
        list(proper.values()),
        list(proper.keys()),
        force_ascii=False,
    )
    fig.show()

```

BQP Python Execution

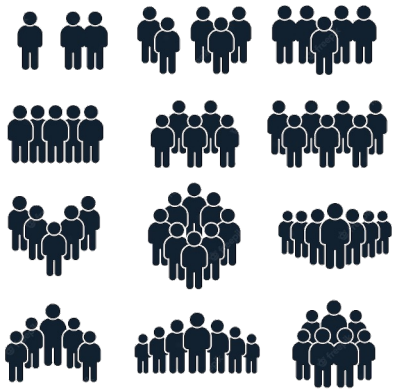
```
BADWLRZ-CM86053:bqp-test di46niz$ python3.10 main_q5_without.py
   q_0: ──X──M──
   q_1: ──X──┼──M──
   q_2: ─H─■─■─■──┼──M──
   q_3: ─────X─┼──┼──M──
   q_4: ───────X──┼──┼──M──
   c: 5/══════════╩══╩══╩══╩══
meas: 5/═══════════0──1──2──3──4──
```

```
{'00000': 309,
 '00001': 58,
 '00010': 78,
 '00011': 16,
 '00101': 7,
 '00110': 1,
 '00111': 3,
 '01000': 26,
 '01001': 5,
 '01010': 5,
 '01011': 1,
 '01100': 2,
 '01101': 10,
 '01110': 2,
 '01111': 19,
 '10000': 15,
 '10001': 6,
 '10010': 4,
 '10011': 2,
 '10100': 8,
 '10101': 19,
 '10110': 2,
 '10111': 33,
 '11000': 3,
 '11001': 6,
 '11011': 2,
 '11100': 22,
 '11101': 128,
 '11110': 34,
 '11111': 174}
```

```
BADWLRZ-CM86053:bqp-test di46niz$
```

HPC Access

Wide HPC User Communities



Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

HPC System With Quantum Accelerator

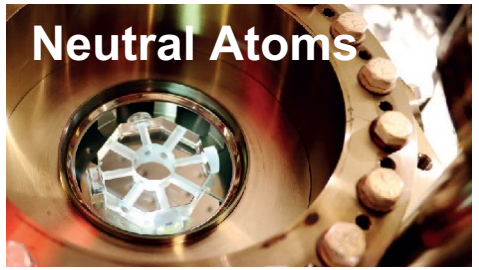
Portal for individual experiments

- Authentication and authorization
- Web-based access
- Simple, but high latency

Access via HPC System

- Submission to HPC scheduler as regular HPC job
- Access to QC system from HPC job
- Offload / Host-based interfaces
  - E.g., OpenMP target

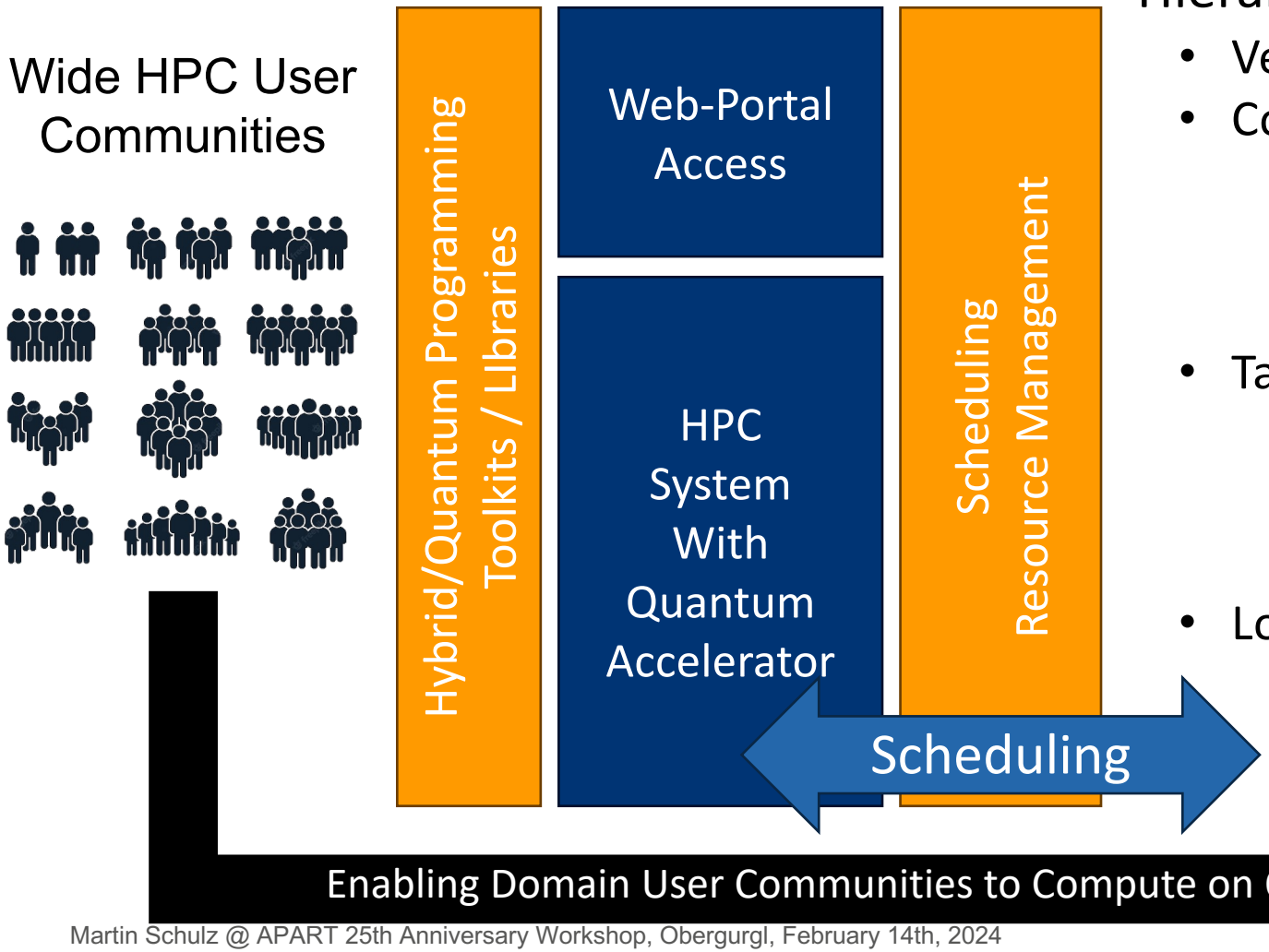
In both cases: result is a templated circuit to be executed



Enabling Domain User Communities to Compute on Quantum Devices

QCs & Simulators

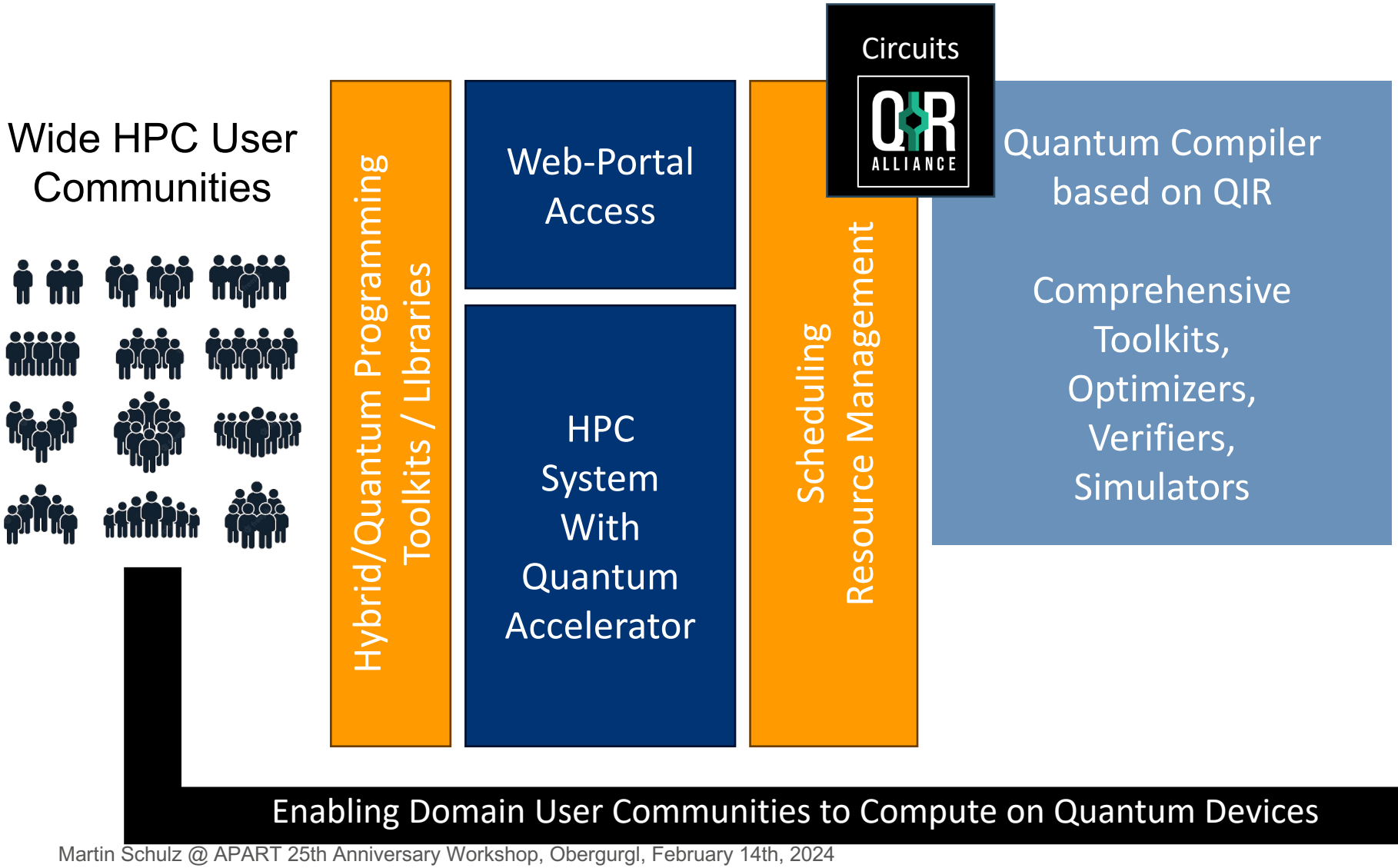
Scheduling & Resource Management



Hierarchical Scheduling Component

- Vendor neutral, open source
- Combining/Coordinating
  - HPC Scheduler (given by center)
  - Device Scheduler (intermediary)
  - Platform Scheduler (given by vendor)
- Target platform selection
  - Simulator vs. Hardware Device
  - Type of modality
  - Suitable device within modality
- Looking at new scheduler technologies
  - OAR (Grenoble)
  - Flux (LLNL)

Quantum Compiler



Modern Compiler

- Based on QIR/LLVM
- Open & Extensible

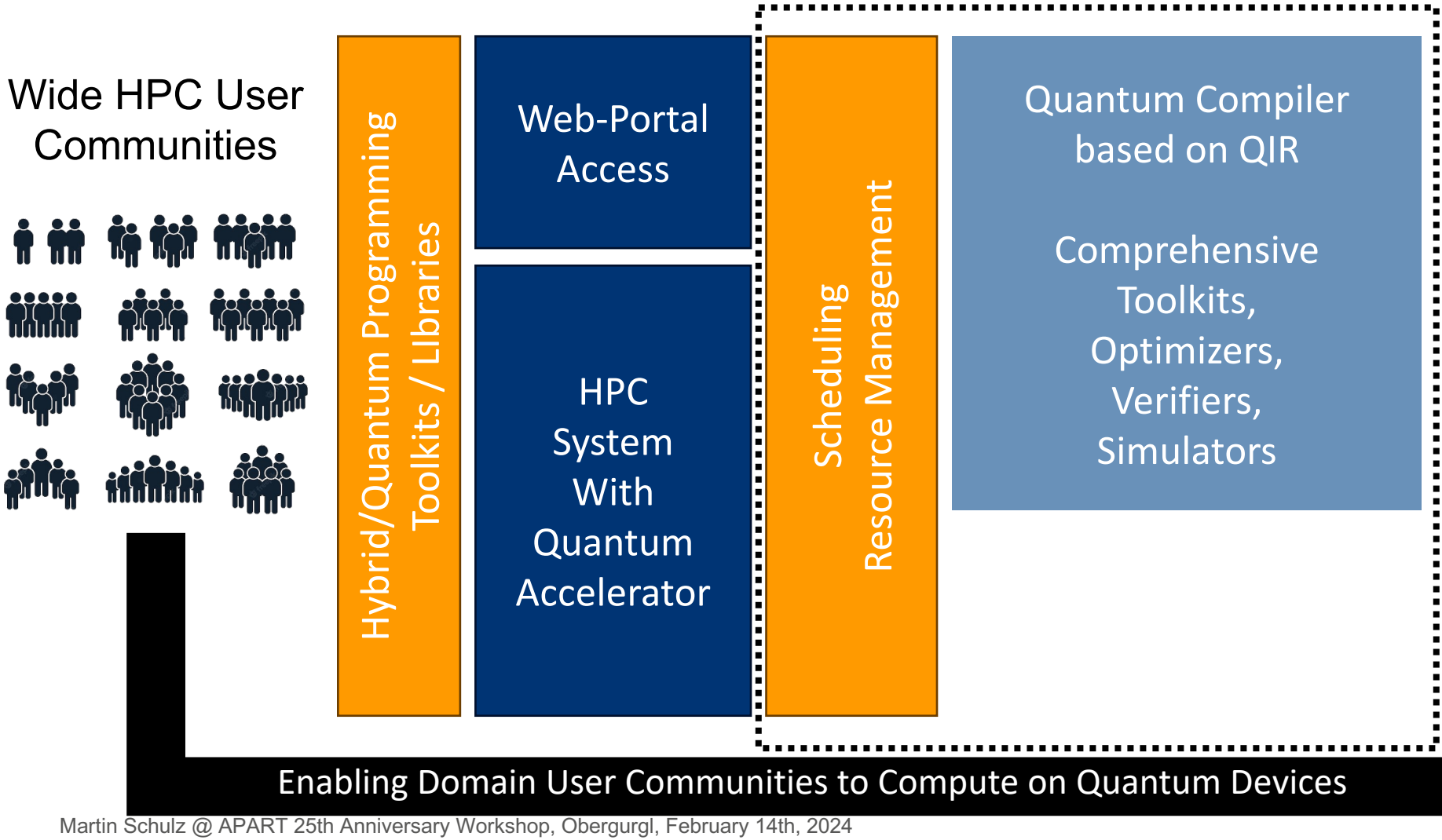
Two main phases

- Platform agnostic
- Platform specific

Each phase has passes

- Transpilers
- Optimizers
- Ion-specific adaptation
- Tools (like MQT)

Quantum Resource Manage



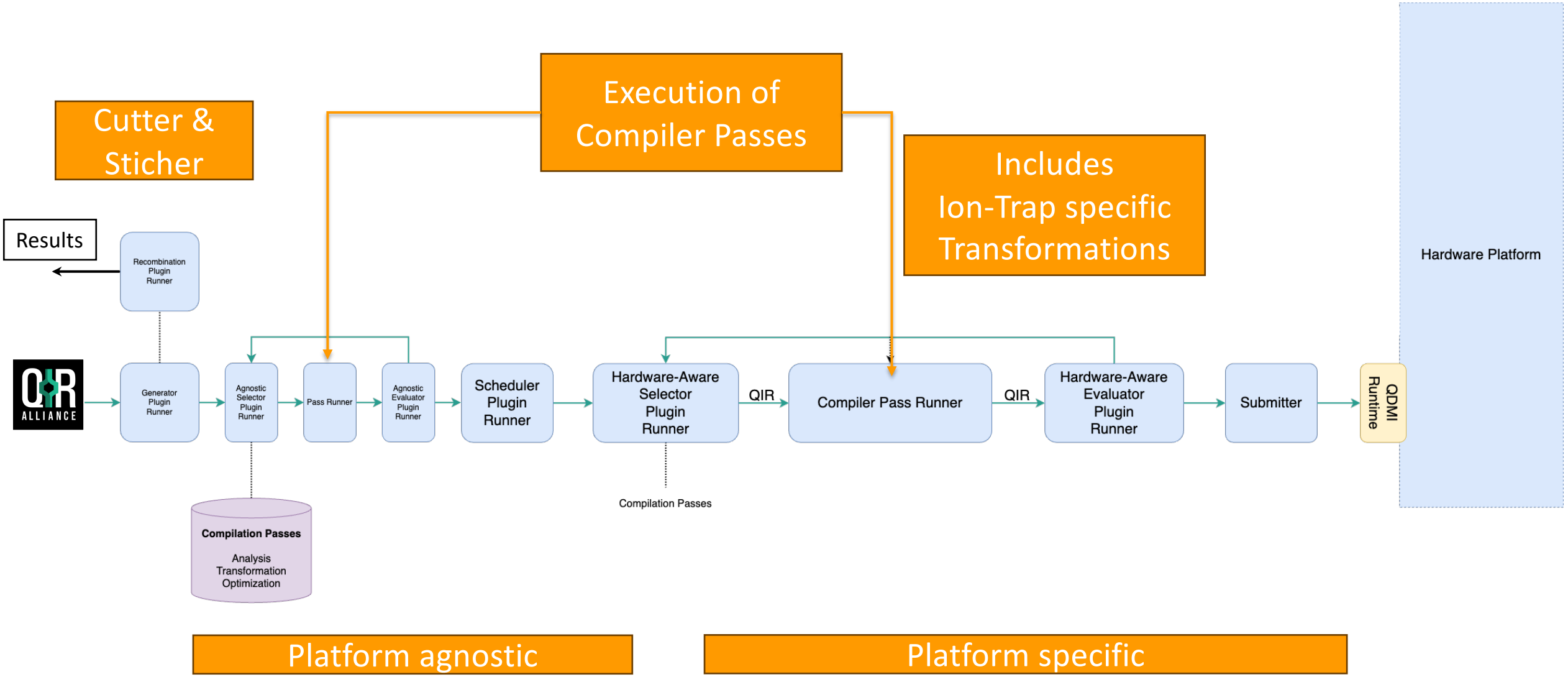
Quantum Resource Manager = QRM

Single process

- Running the framework
- Extensible with plugins (passes and tools)
- Open interfaces
- HPC capable
- Well defined backend interface

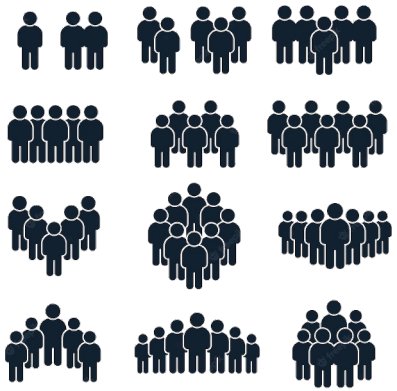
Compilation Process in the MQSS

Driving an LLVM-based open compilation infrastructure



The QDMI Backend

Wide User Communities



Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

HPC System With Quantum Accelerator

Scheduling Resource Management

Quantum Compiler based on QIR

Comprehensive Toolkits, Optimizers, Verifiers, Simulators

QDMI = Quantum Backened Interface

QDMI = Single, Open, Standardized Backend Interface

Three components

- Execution
- Query
- Cailbration (SW trigger)

Enabling Domain User Communities to Compute on Quantum Devices

QCs & Simulators

QDMI Basics

C-API to access QC systems from compilation backend (QDMI Control)

- API is Platform neutral
- Core implementation maps API to a backend plugin

QDMI Core

Find and Instantiate Devices

```
/* Core Session, basic start/close routines */


int QDMI_session_init(QInfo info, QDMI_Session *session);
int QDMI_session_finalize(QDMI_Session session);


/* Core Interface */

int QDMI_core_version(QDMI_Session *session, int* major, int* minor);
int QDMI_core_device_count(QDMI_Session *session, int *count);
int QDMI_core_open_device(QDMI_Session *session, int idx, QInfo *info,
     QDMI_Device* handle);
int QMPI_core_query_device(QDMI_Session *session, int idx, QInfo *info);
int QDMI_core_close_device(QDMI_Session *session, QDMI_Device handle);
```

QDMI Basics

C-API to access QC systems from compilation backend (QDMI Control)

- API is Platform neutral
- Core implementation maps API to a backend plugin
- System specific backends
- Remote forwarding option for systems not located at LRZ
  (higher latency, loose coupling, but access and execution possible)

QDMI Control

Create, Submit and Control QC jobs, Read Results

```
int QDMI_control_pack_qasm2(QDMI_Device dev, char *qasmstr, QDMI_Fragment *frag);
int QDMI_control_pack_qir(QDMI_Device dev, void *qirmod, QDMI_Fragment *frag);
int QDMI_control_submit(QDMI_Device dev, QDMI_Fragment *frag, int numshots, QInfo int
QDMI_control_cancel(QDMI_Device dev, QDMI_Job *job, QInfo info);
int QDMI_control_pause(QDMI_Device dev, QDMI_Job *job, QInfo info);
int QDMI_control_test(QDMI_Device dev, QDMI_Job *job, int *flag, QDMI_Status *status);
int QDMI_control_wait(QDMI_Device dev, QDMI_Job *job, QDMI_Status *status);
int QDMI_control_extract_state(QDMI_Device dev, QDMI_Status status, int *state);
int QDMI_control_readout_size(QDMI_Device dev, QDMI_Status *status, int *numbits);
int QDMI_control_readout_hist_size(QDMI_Device dev, QDMI_Status *status, int *size);
int QDMI_control_readout_hist_top(QDMI_Device dev, QDMI_Status *status, int numhist, QInfo
                info, long *hist);
int QDMI_control_readout_raw_num(QDMI_Device dev, QDMI_Status *status, int *num);
int QDMI_control_readout_raw_sample(QDMI_Device dev, QDMI_Status *status, int numraw, QInfo
                info, long *hist);
```

QDMI Basics

C-API to access QC systems from compilation backend (QDMI Control)

- API is Platform neutral
- Core implementation maps API to a backend plugin
- System specific backends
- Remote forwarding option for systems not located at LRZ
  (higher latency, loose coupling, but access and execution possible)

API for functionality to read dynamic data from QC system (QDMI Query)

- Enable proper selection of suitable backends
- Usage by compilation passes to drive optimizations

QDMI Query

Ask for system properties, gate sets and gate properties (e.g., fidelities)

```
int QDMI_query_device_property_exists(QDMI_Device dev, QDMI_Device_property prop, int* scope);
int QDMI_query_device_property_i(QDMI_Device dev, QDMI_Device_property prop, int *value);
int QDMI_query_device_property_f(QDMI_Device dev, QDMI_Device_property prop, float *value);
int QDMI_query_device_property_d(QDMI_Device dev, QDMI_Device_property prop, double *value);

int QDMI_query_gateset_num(QDMI_Device dev, int *num_gates);
int QDMI_query_one_gate(QDMI_Device dev, QDMI_Gate *gate);
int QDMI_query_all_gates(QDMI_Device dev, QDMI_Gate *gates);
int QDMI_query_byname(QDMI_Device dev, char *name, QDMI_Gate *gate);

int QDMI_query_gate_name(QDMI_Device dev, QDMI_Gate gate, char* name, int* len);
int QDMI_query_gate_size(QDMI_Device dev, QDMI_Gate gate, int* gatesize);
int QDMI_query_gate_unitary(QDMI_Device dev, QDMI_Gate gate, QDMI_Unitary *unitary);

int QDMI_query_gate_property_exists(QDMI_Device dev, QDMI_Gate gate, QDMI_Gate_property prop, int* scope);
int QDMI_query_gate_property_i(QDMI_Device dev, QDMI_Gate_property prop, QDMI_Gate gate, int *coor, int*
                  value);
int QDMI_query_gate_property_f(QDMI_Device dev, QDMI_Gate_property prop, QDMI_Gate gate, int *coor, float*
                  value);
int QDMI_query_gate_property_d(QDMI_Device dev, QDMI_Gate_property prop, QDMI_Gate gate, int *coor, double*
                  value);
```

QDMI Basics

C-API to access QC systems from compilation backend (QDMI Control)

- API is Platform neutral
- Core implementation maps API to a backend plugin
- System specific backends
- Remote forwarding option for systems not located at LRZ
  (higher latency, loose coupling, but access and execution possible)

API for functionality to read dynamic data from QC system (QDMI Query)

- Enable proper selection of suitable backends
- Usage by compilation passes to drive optimizations

API to Access to regular re-calibration (QDMI Device)

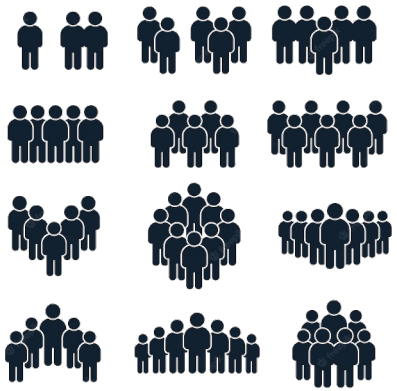
QDMI Device

Access to re-calibration

```
int QDMI_device_status(QDMI_Device dev, QInfo info, int *status);
int QDMI_device_quality_check(QDMI_Device dev, double *result);
int QDMI_device_quality_limit(QDMI_Device dev, double *result);
int QDMI_device_quality_calibrate(QDMI_Device dev);
```

End-to-End to the Systems

Wide User Communities



Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

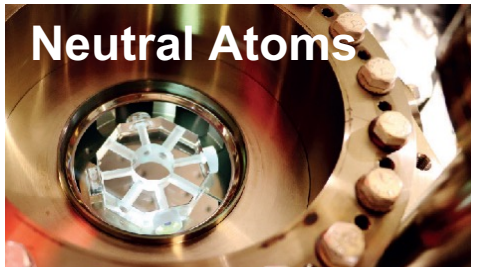
HPC System With Quantum Accelerator

Scheduling Resource Management

Quantum Compiler based on QIR

Comprehensive Toolkits, Optimizers, Verifiers, Simulators

QDMI = Quantum Backened Interface



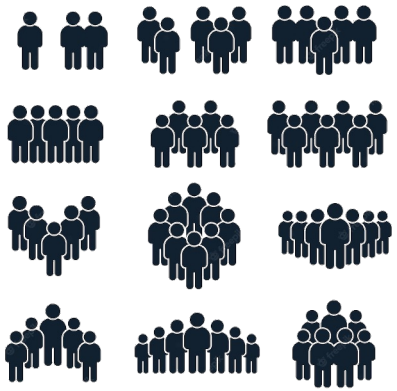
Enabling Domain User Communities to Compute on Quantum Devices

QCs & Simulators

Feedback from the Target System

Adding Feedback into Compilation Process

Wide User Communities



Hybrid/Quantum Programming Toolkits / Libraries

Web-Portal Access

HPC System With Quantum Accelerator

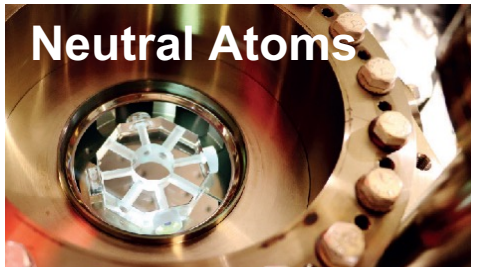
Scheduling Resource Management

Quantum Compiler based on QIR

Comprehensive Toolkits, Optimizers, Verifiers, Simulators

Figures of Merit and Constraints

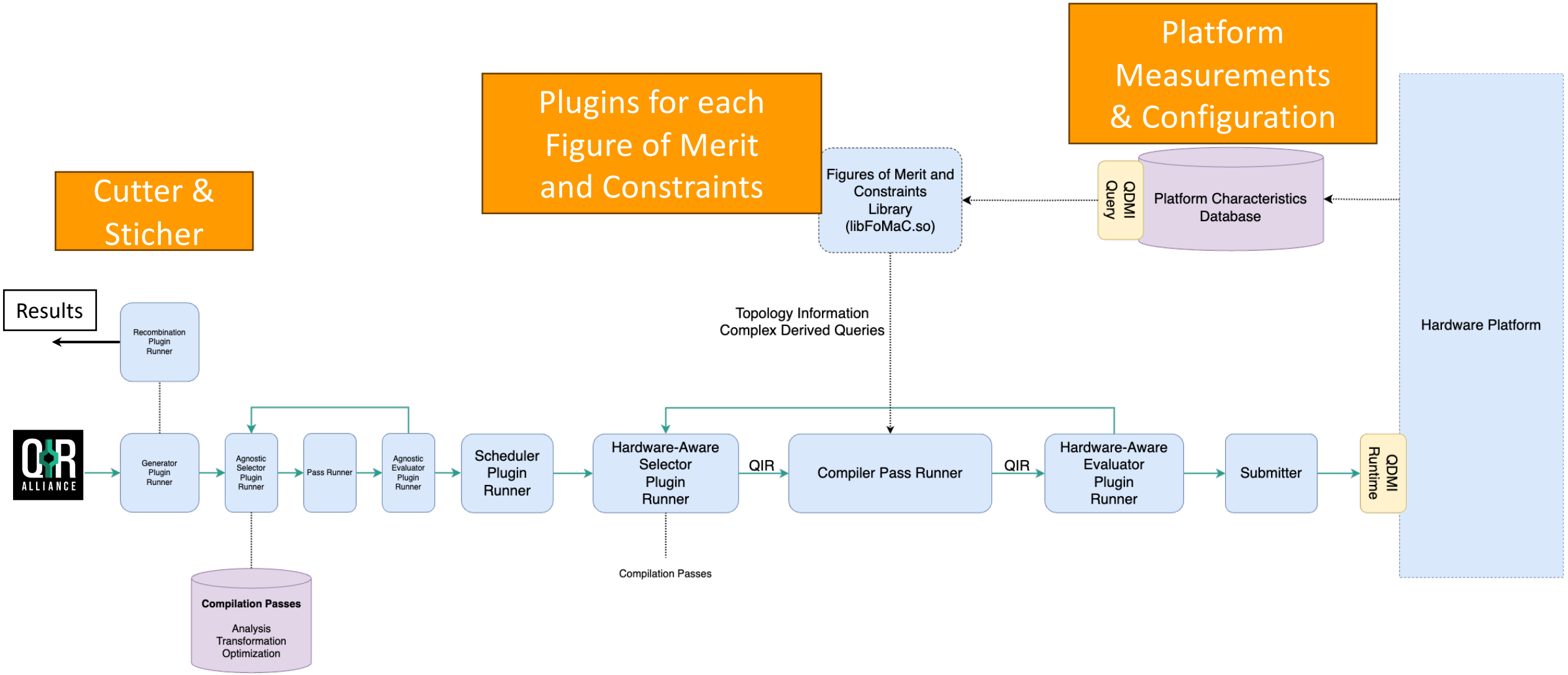
QDMI = Quantum Backened Interface



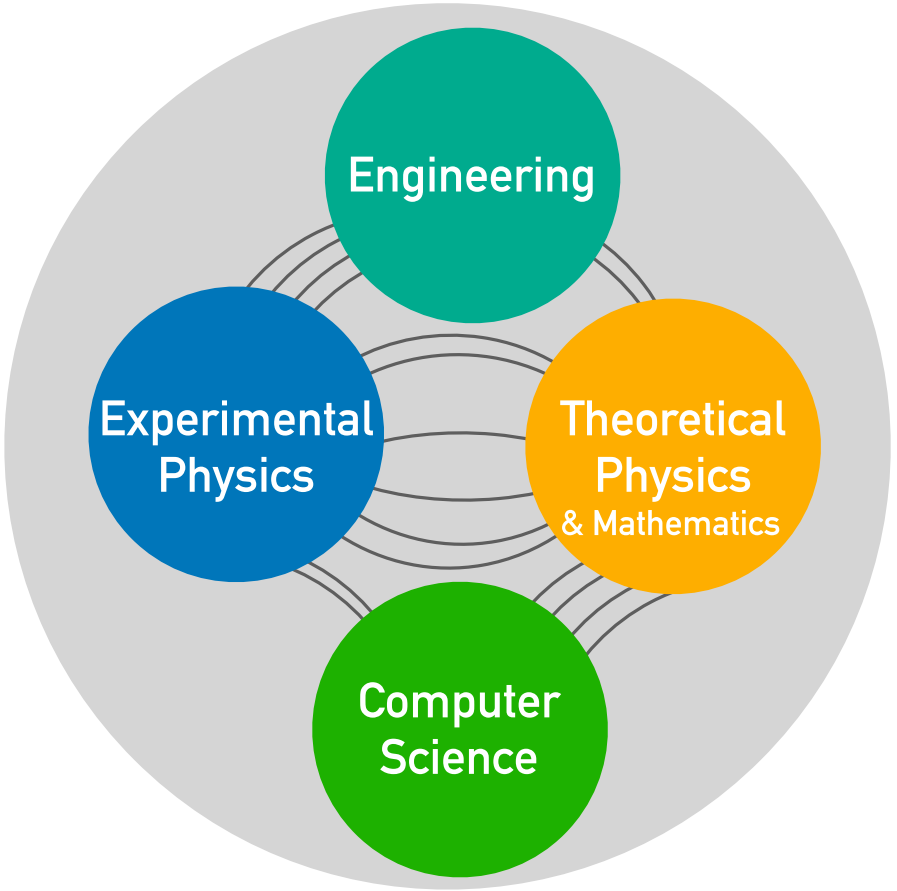
Enabling Domain User Communities to Compute on Quantum Devices

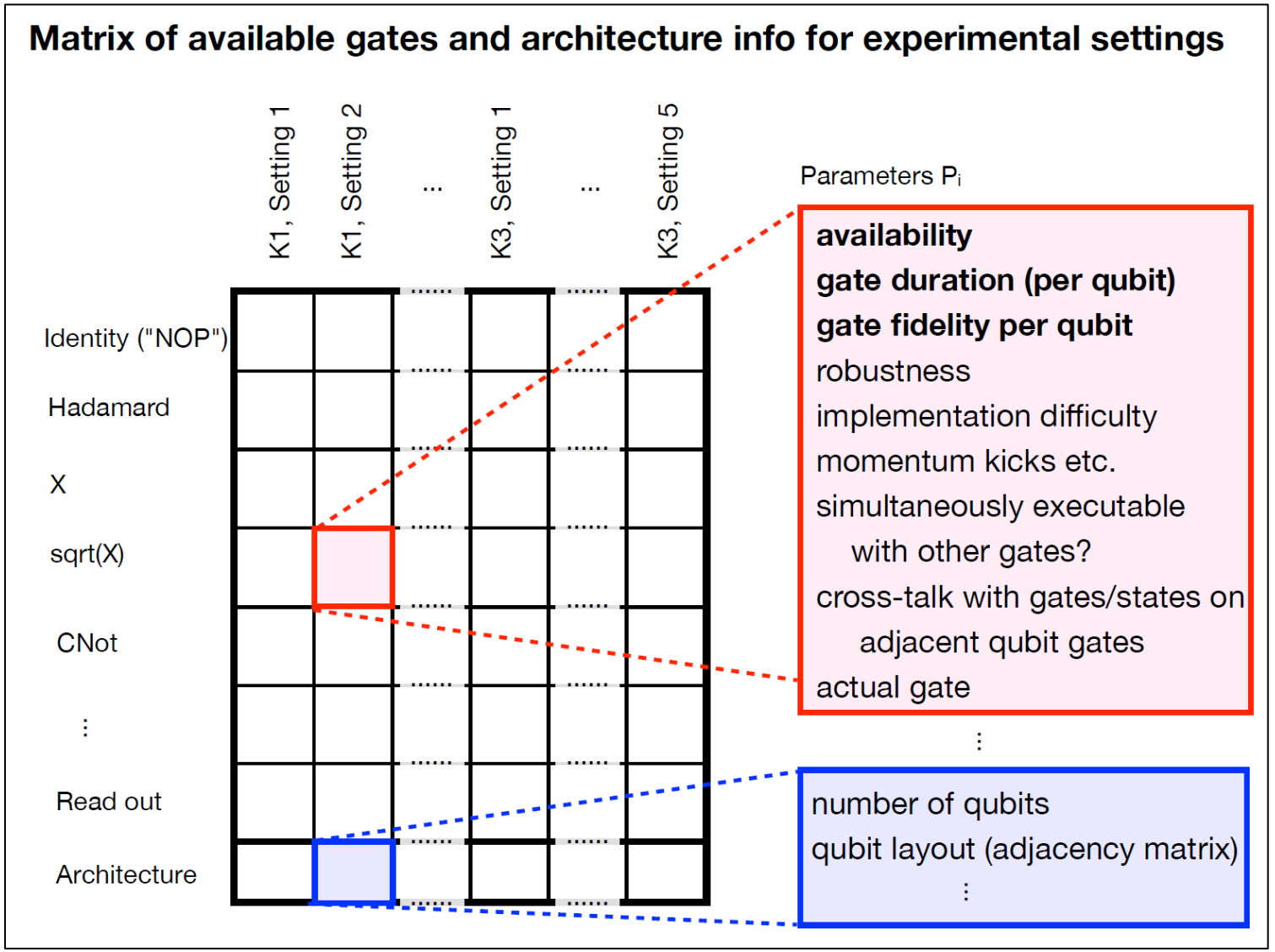
QCs & Simulators

Driving an LLVM-based open compilation infrastructure



Representing Figures of Merit



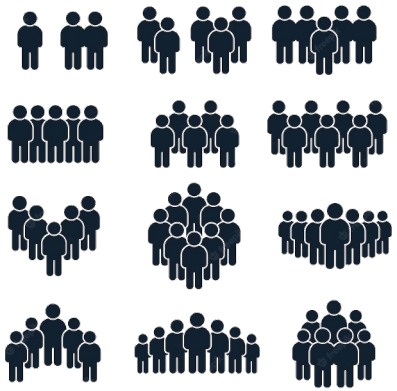


Separating Concerns

Plattform Component

Open Infrastructure

Wide User Communities



id/Quantum Programming Toolkits / LIbraries

Ion-specific Abstraction (optional)

Web-Portal Access

HPC System With Quantum Accelerator

Scheduling Resource Management

Selector Plugin

Quantum Compiler based on QIR

Comprehensive Toolkits,

Compiler Pass

Mapping To Pulses

s of Merit straints

FoMaC Calc.

QDMI = Quantum Backened Interface

Backend Impl.

W-Emulation

Machine

Backend Impl.

Ion Traps

CCD

Neutral Atoms

Backend Impl.

Enabling Domain User Communities to Compute on Quantum Devices

QCs & Simulators

The Munich Quantum Software Stack (MQSS)

Architecture Mapping to Hardware

Compute Node

Service Node

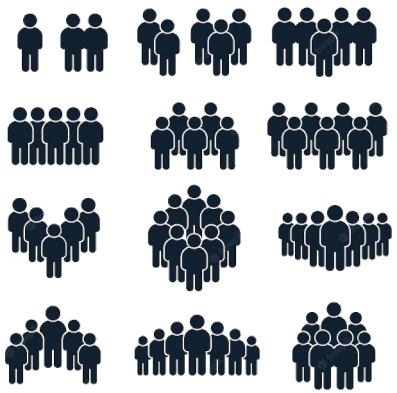
Frontend Node

QC System

QC Control

Dedicated Ethernet

Wide User Communities



Web-Portal Access

HPC Scheduler (HPC center controlled)

Hybrid/Quantum Programming

Application Process

HPC Application Process With Quantum Code

Scheduling Resource Management

Quantum Compiler based on QIR

Comprehensive Toolkits, Optimizers, Verifiers, Simulators

QDMI

Single process with QRM & QDMI libraries linked

Enabling Domain User Communities to Compute on Quantum Devices

QCs & Simulators

Some HPCQC End Goals

Infrastructure	Standalone system with own monitoring	→	Integrated in data center with cooling, power/energy, monitoring
Hardware	Quantum Computing Appliance + HPC System	→	Single system with integrated QC accelerator(s)
Software	Two separate software stacks with “glue scripts”	→	Single-source programming with high-level abstractions
Applications	Algorithms with quadratic speedup and separate QC and HPC applications	→	Algos with faster speedup and truly hybrid applications
Culture	Computer Science, Math, Electrical Engineering, Physics communities	→	Hybrid HPCQC community


European Quantum
SYSTEM SOFTWARE SUMMIT
European Quantum System Software Summit
April 19-21, 2023 Raitenhaslach
European Quantum
SYSTEM SOFTWARE SUMMIT

ware Summit

European Quantum
SYSTEM SOFTWARE SUMMIT

European Quantum System Software Summit

July 11-13, 2023 LRZ

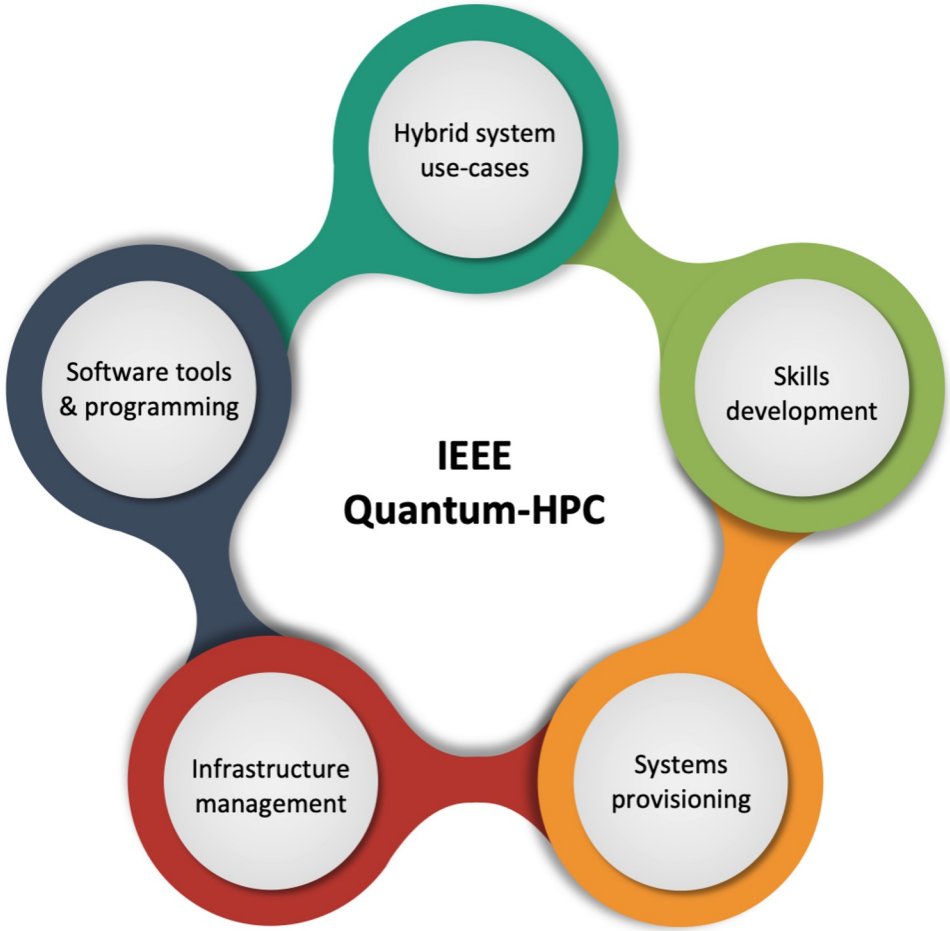
IEEE QUANTUM

Quantum-HPC Working Group

Building a global community for sustained, structured engagements

Developing community-driven best practices for deploying, operating and using hybrid HPC-QCS systems, tools and applications

- Launched at IEEE QCE23 (September 2023)
- SC23 BOF Session (November 2023)
- More BoFs/Events/Workshops planned



ICHEC Irish Centre for High-End Computing

Los Alamos NATIONAL LABORATORY

OAK RIDGE National Laboratory

Technische Universität München

Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities

Read more at quantum.ieee.org/working-groups

or grab the QR code

It is a Software Challenge, too!

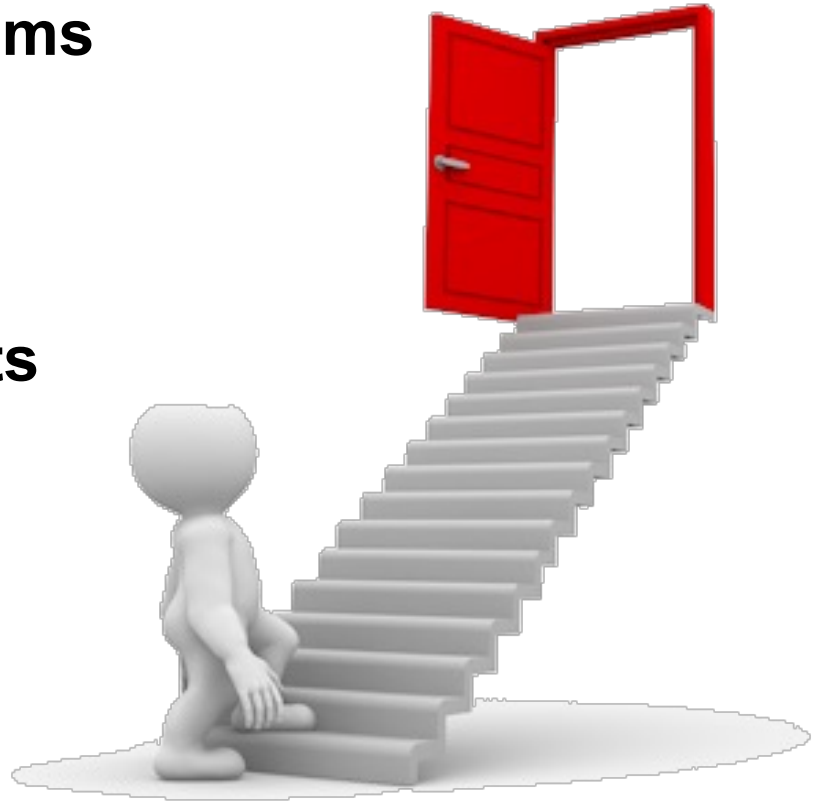
A Comprehensive Software Stack is Key for QC Systems

- Enables wide range of user communities
- Delivers efficient workflows from code to optimized pulses
- Provides safe production-level multi-user operations

QC must be integrated into existing HPC environments

- Targets specific computational problems
- Is part of larger hybrid workflows
- Relies on HPC resources for optimization

➢ Single stack build on top of established HPC technology

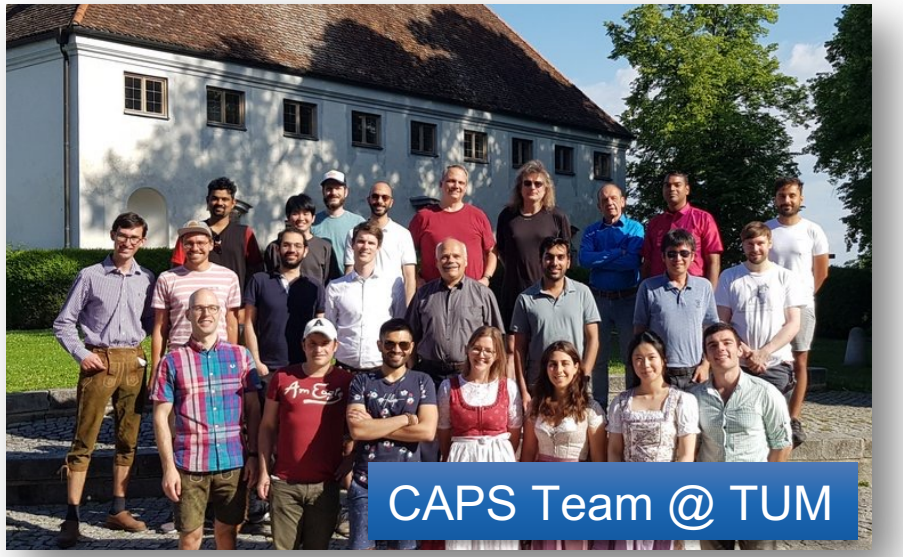


The Munich Quantum Computing Software Stack

- Portal and HPC access to heterogeneous backends
- New programming models for direct access
- Efficient compilers and optimizers
- Comprehensive tools for verification and optimization

Acknowledgements

It takes a team, or rather many teams!



CAPS Team @ TUM

QCT Team @ LRZ

From the MQV Review Meeting 2022

Bayerisches Staatsministerium für Wissenschaft und Kunst

Federal Ministry of Education and Research

Federal Ministry for Economic Affairs and Climate Action

EuroHPC Joint Undertaking

European Commission

QC@LR HPCQC

Towards Practical Quantum Computing

It is a Software Challenge, too!

Munich Quantum Valley

Q-DESSI

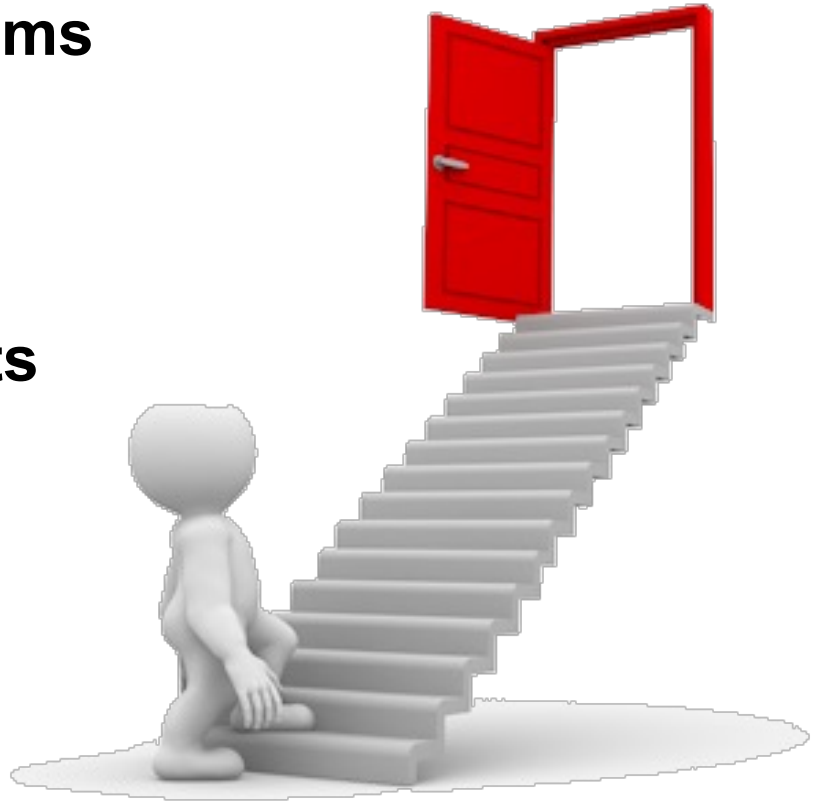
A Comprehensive Software Stack is Key for QC Systems

- Enables wide range of user communities
- Delivers efficient workflows from code to optimized pulses
- Provides safe production-level multi-user operations

QC must be integrated into existing HPC environments

- Targets specific computational problems
- Is part of larger hybrid workflows
- Relies on HPC resources for optimization

➢ Single stack build on top of established HPC technology



The Munich Quantum Computing Software Stack

- Portal and HPC access to heterogeneous backends
- New programming models for direct access
- Efficient compilers and optimizers
- Comprehensive tools for verification and optimization

QC@LRZ HPCQC MQT