www.q-solid.de

Quantum Computer in the Solid State

JUNIQ Cloud Provides Access to Quantum Computer Emulator JUQCS

Carlos Gonzalez Calaza [1,2], Cong Luo [1], Hans De Raedt [3], Kristel Michielsen [1,2]

[1] Institute for Advanced Simulation, Jülich Supercomputing Centre, Forschungszentrum Jülich, D-52425 Jülich, Germany

[2] RWTH Aachen University, D-52056 Aachen, Germany

[3] Zernike Institute for Advanced Materials, University of Groningen, Nijenborgh 4, NL-9747 AG Groningen, The Netherlands

Background

The Jülich UNified Infrastructure for Quantum computing (JUNIQ) [1] is a uniform quantum computing Platform as a Service (QC-PaaS) operated at the Jülich Supercomputing Centre (JSC). We present the architecture and usage of qiskit-juqcs [2], a Python library designed to provide users of JUNIQ to access the Jülich Universal Quantum Computer Simulator (JUQCS) [3], a massively parallel emulator of gate-based quantum computers which has set the world record of simulating a universal quantum computer with 48 qubits. This work enables QC users with no High Performance Computing (HPC) experience to easily simulate large quantum computing circuits remotely on HPC systems through the widely used Qiskit interface.

Architecture

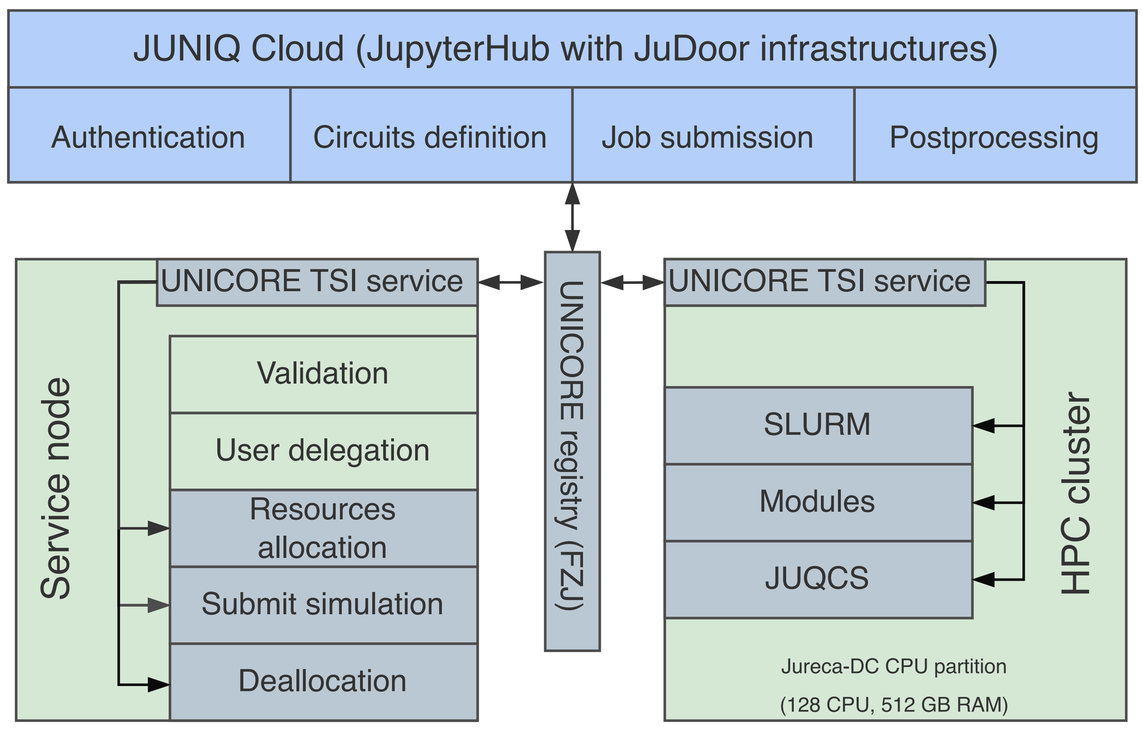

Fig. 1 Architecture of the cloud access to JUQCS via JUNIQ.

JUNIQ cloud is a JupyterHub environment, users can register via JuDoor [4]. Transfer of input and output data from/to JUNIQ cloud and execution of the workflow on HPC system is enabled by the grid middleware technology UNICORE (Uniform Interface to Computing Resources) [5]. The service node between JUNIQ cloud and HPC system is responsible for user authentication, job preparation and submission, and result retrieval.

Interfacing with Qiskit

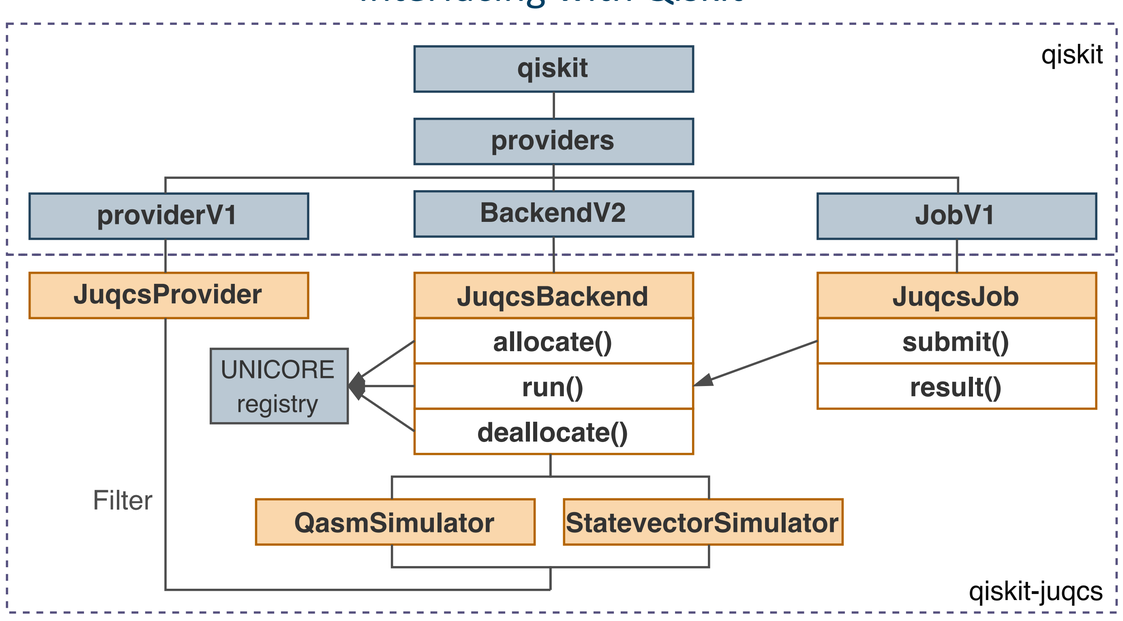

Fig. 2 Interfacing the JUNIQ cloud and JUQCS with Qiskit. Notably, three new methods for the class JuqcsBackend (derived from class BackendV2 that Qiskit provided) have been created to interact with UNICORE, hence leveraging the architecture.

Demonstration

Importing / Authentication

```
import qiskit
from juqcs.provider import JuqcsProvider
```

```
Credentials are valid! You may start using JUQCS now.
```

```
nqubits = 40
circuit = qiskit.QuantumCircuit(nqubits)
circuit.h(0)
for i in range(1, nqubits):
    circuit.cx(i - 1, i)
circuit.measure_all()
```

Provider / Backend instantiation, resource allocation

```
provider = JuqcsProvider()
backend = provider.get_backend('qasm_simulator')
backend.allocate(minutes=30, max_qubits=nqubits)
```

```
Please choose one of the following projects to charge the allocation to: ['jqctst
02', 'jqctst01']. Get rid of this prompt by actively passing a project string to
the "allocate" method. jqctst01
Trying to allocate compute resources under project jqctst01, this may take a few
minutes... (please do not abort, otherwise the allocation may be lost and the com
pute time wasted).
Resource allocation #12359303 of 40 qubits available until 2023-10-13 12:51:05.
```

Circuit submission

```
job = qiskit.execute(circuit, backend, shots=5000, seed=42)
```

```
Submitting circuits for simulation, this may take a few minutes...
```

```
job.status()
```

```
<JobStatus.DONE: 'job has successfully run'>
```

```
result = job.result()
result.get_counts()
```

```
{'11111111111111111111111111111111111111111': 2471,
 '00000000000000000000000000000000000000000': 2529}
```

Resource deallocation

```
backend.deallocate()
```

```
Trying to deallocate compute resources, this may take a few minutes...
Allocation #12359303 revoked.
```

Fig. 3 Demonstration of accessing JUQCS via JUNIQ cloud, framed cells indicate the key usage paradigm of qiskit-juqcs. In this example, we submit a circuit consisting of 40 qubits with 5000 shots using the qasm_simulator backend running on 128 compute nodes (JURECA-DC CPU partition).

Acknowledgements:

We thank Fengping Jin, Piet Hein van den Heuvel, Dennis Willsch and Jhon Alejandro Montanez-Barrera for the contributions to the design of the architecture, as well as for the thorough testing of the implementation shown in this poster. C.G.C and C.L. acknowledge support from the project JUNIQ that has received funding from the German Federal Ministry of Education and Research (BMBF) and the Ministry of Culture and Science of the State of North Rhine-Westphalia, as well as the project QSolid that has received funding from the German Federal Ministry of Education and Research (BMBF) within the framework programme “Quantum technologies – from basic research to market”. C.G.C. acknowledges support from the project OpenSuperQ (820363) of the European Quantum Flagship.

References:

[1] JUNIQ: https://juniq.fz-juelich.de

[2] qiskit-juqcs: https://jugit.fz-juelich.de/qip/juniq-platform/qiskit-juqcs/

[3] H. De Raedt et al., Massively parallel quantum computer simulator, eleven years later, Comput. Phys. Commun. 237, 47 (2019)

[4] JuDoor: judoor.fz-juelich.de/

[5] UNICORE: https://www.unicore.eu/

SPONSORED BY THE

Federal Ministry of Education and Research

The QSolid project acknowledges the support of the Federal Ministry of Education and Research (BMBF) within the framework programme “Quantum technologies – from basic research to market” (Grant No. 13N16149)