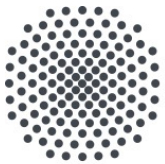
Quantum Applications in the NISQ Era



University of Stuttgart

Marie Salm

salm@iaas.uni-stuttgart.de

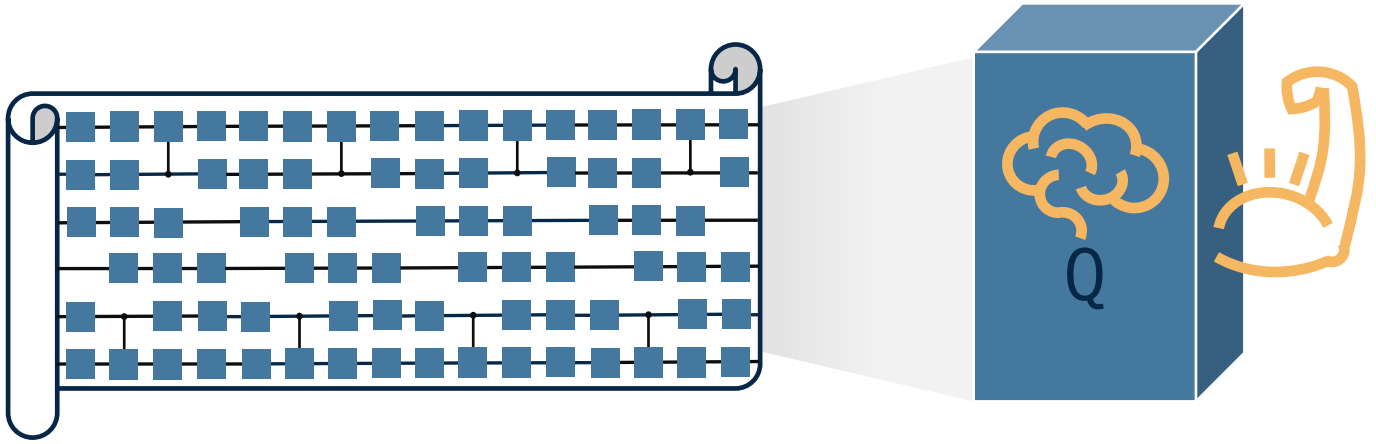
Institute of Architecture of Application Systems

Supported by:

Federal Ministry
for Economic Affairs
and Climate Action

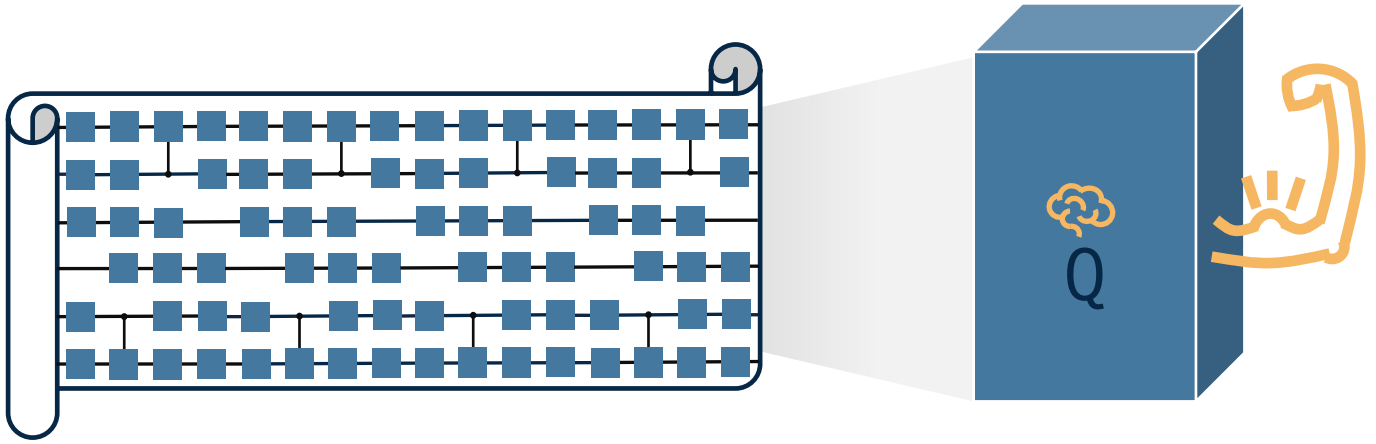
on the basis of a decision
by the German Bundestag

Future vs. today



- QPUs will solve problems faster

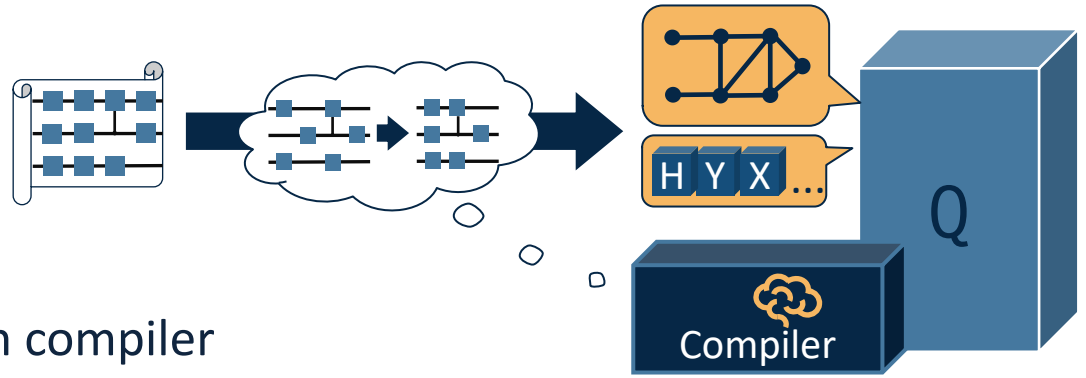
Future vs. today



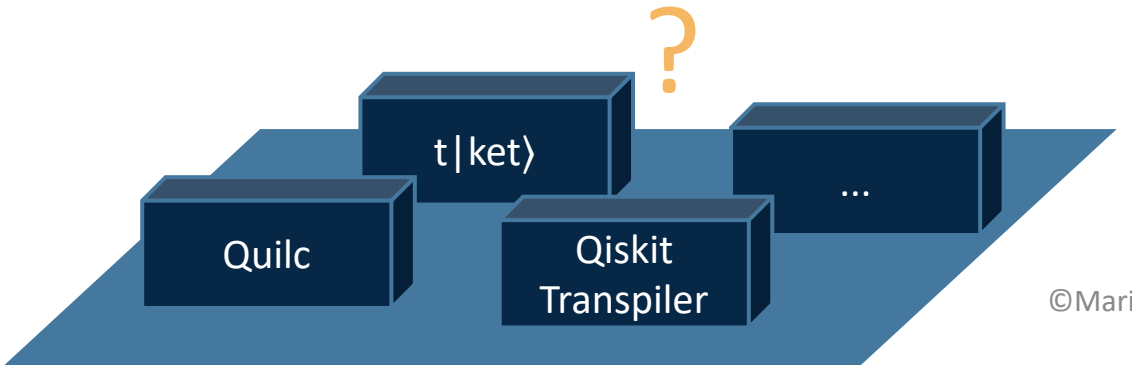
- QPUs will solve problems faster
- **currently**: prone to errors & small qubit counts
  - determine **executable circuit sizes**, i.e., width & depth
  - design quantum circuits as **resource efficient** as possible

Influencing circuit size

- **true size** of circuit depends on
  - qubit topology
  - implemented gate set
  - mapping & optimization by quantum compiler

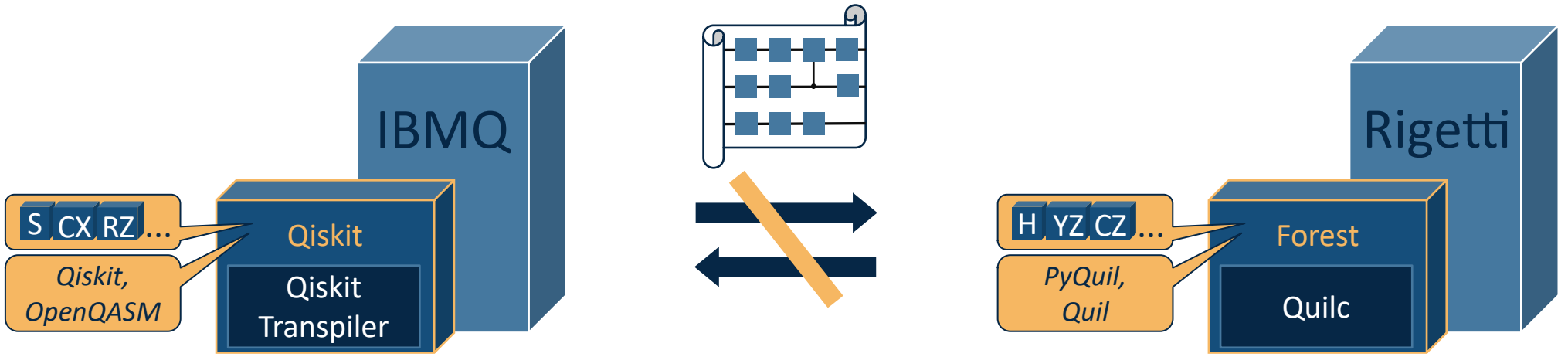


- **variety** of quantum compilers exists
- given a circuit: which compiler delivers "**best" compilation result**?
- comparison is required ...



Usage of quantum compilers

- most compilers **accessible via SDKs** differing in
  - gate sets
  - programming languages
  - vendors of QPUs



Analysis of quantum compilers and their SDKs [1]

- **not** every SDK supports OpenQASM
- many vendor-specific SDKs only support **own QPUs**
- many SDKs support custom QPUs but **not custom target gate sets**
  - formats for defining custom QPUs differ
  - ➢ SDK has to support the target QPU natively

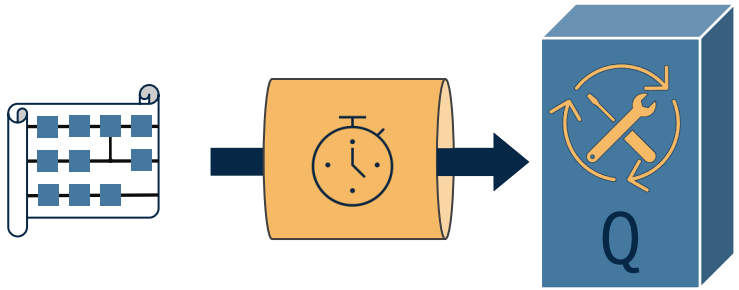
	SDKs (Compilers)	Import Languages	Export Languages	Backend Vendors	Custom QPUs	Custom Compilation Gate Set
Propr. SDKs	Cirq	OpenQASM, Cirq-JSON, Quirk-JSON	OpenQASM, Quil, Cirq-JSON	Google, AQT, Pasqal	yes	yes
	Forest (Quilc)	PyQuil*, Quil	Quil	Rigetti	yes	no
	Qiskit (Transpiler)	OpenQASM, Qiskit*	OpenQASM	IBM	yes	yes
Independent SDKs	pytket (t|ket⟩)	OpenQASM, Qiskit*, PyZX*, PyQuil*, Cirq*, Quipper	OpenQASM, Qiskit*, PyZX*, PyQuil*, Cirq*, Qulacs*, Q#, ProjectQ*	AQT, Amazon Braket, Honeywell, Rigetti, IBM, Microsoft QDK	yes	yes
	staq	OpenQASM	OpenQASM, ProjectQ*, Quil, Q#, Cirq*	no	no	no
	ProjectQ	ProjectQ*	ProjectQ*	IBM, AQT, Amazon Braket, IonQ	no	yes

*Python libraries

[1] Marie Salm, Johanna Barzen, Frank Leymann, Benjamin Weder, Karoline Wild. 2021. Automating the Comparison of Quantum Compilers for Quantum Circuits. In: Proceedings of the 15th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2021), Springer International Publishing, 2021

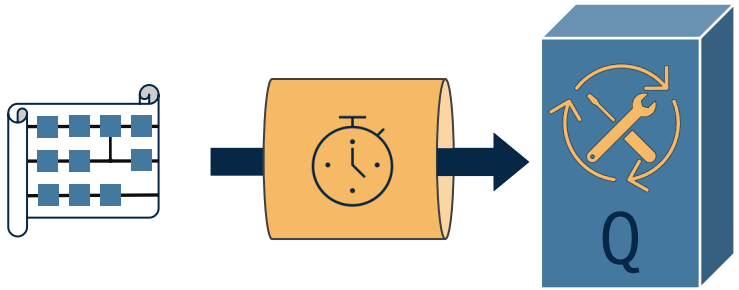
In addition ...

- QPUs are regularly re-calibrated
  - changes in error rates
- cloud providers regulate computing access
  - resulting in waiting times
- implementation & execution of circuit
  - QPU
  - compiler
  - SDK
  - access regulations



In addition ...

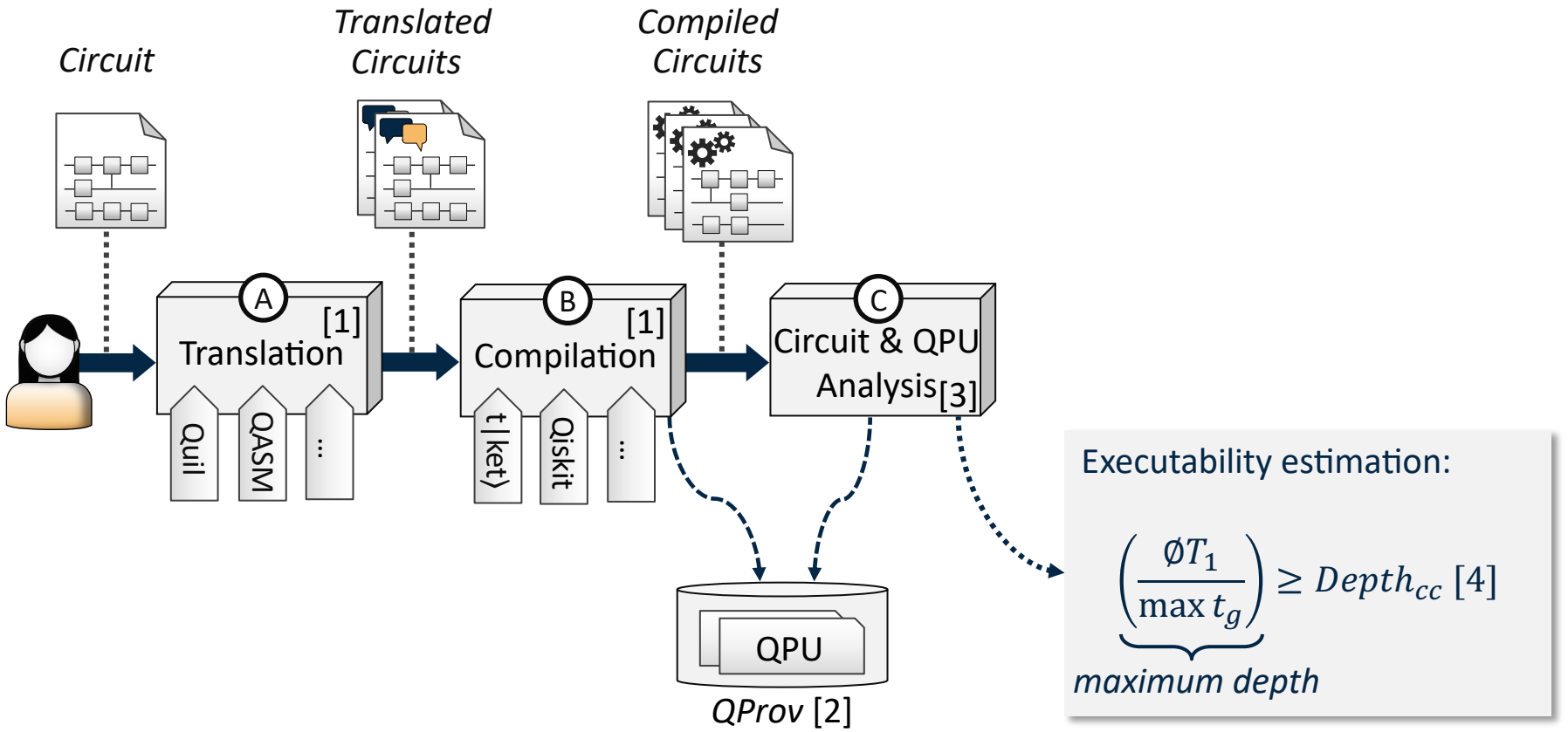
- QPUs are regularly re-calibrated
  - changes in error rates
- computing access
- implementation & ex
  - QPU
  - compiler
  - SDK
  - access regulations



deep **technical knowledge & huge manual effort** required

Approach

Approach – NISQ Analyzer



[1] Marie Salm, Johanna Barzen, Frank Leymann, Benjamin Weder, Karoline Wild. 2021. Automating the Comparison of Quantum Compilers for Quantum Circuits. In: Proceedings of the 15th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2021), Springer International Publishing, 2021

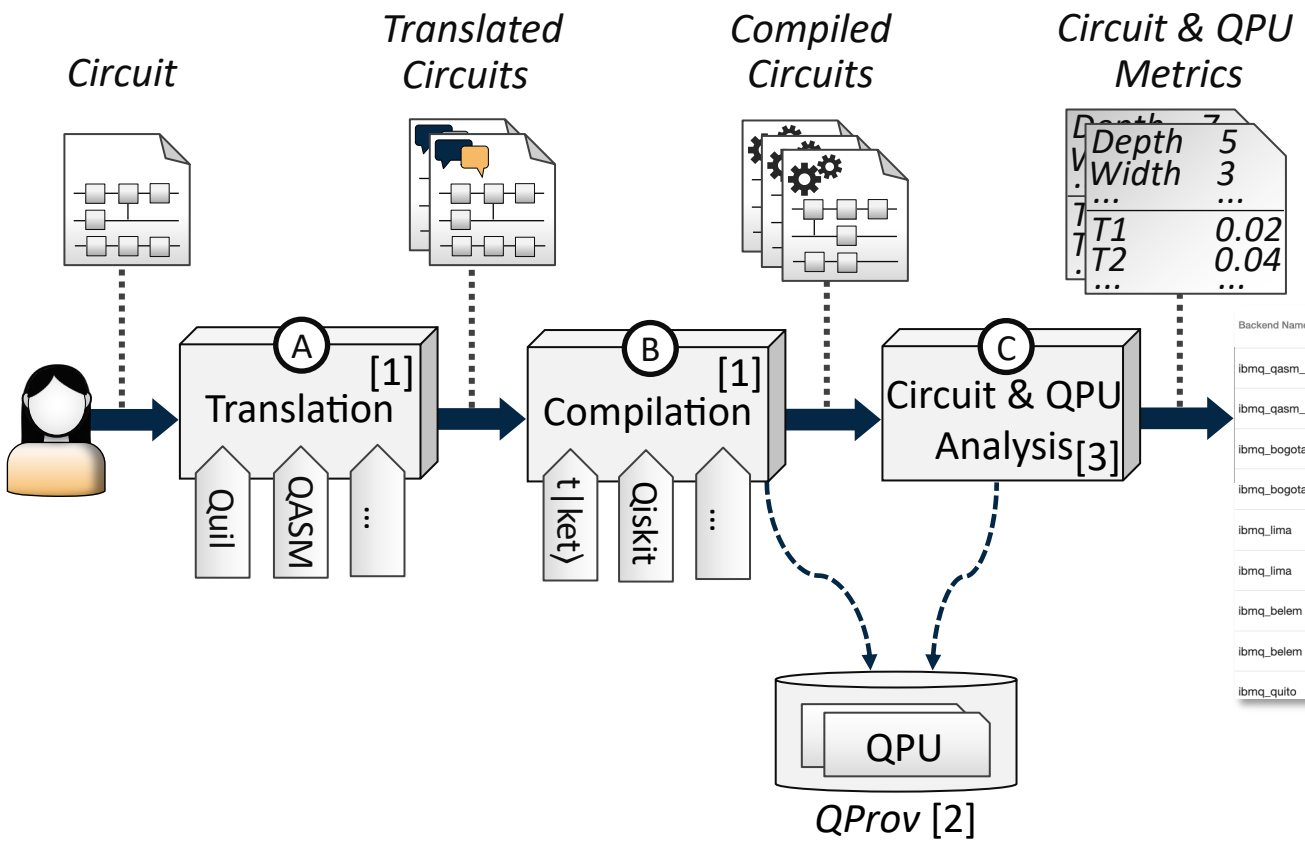
[2] Benjamin Weder, Johanna Barzen, Frank Leymann, Marie Salm, Karoline Wild. 2021. QProv: A provenance system for quantum computing. In: IET Quantum Communications, Wiley, 2021

[3] Marie Salm, Johanna Barzen, Uwe Breitenbücher, Frank Leymann, Benjamin Weder, et al. 2020. The NISQ Analyzer: Automating the Selection of Quantum Computers for Quantum Algorithms. In: Proceedings of the 14th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2020), Springer International Publishing, 2020

[4] Eyob A. Sete, William J. Zeng, and Chad T. Rigetti. 2016. A functional architecture for scalable quantum computing. In 2016 IEEE International Conference on Rebooting Computing(ICRC).1–6.

[5] Marie Salm, Johanna Barzen, Frank Leymann, Benjamin Weder: Prioritization of Compiled Quantum Circuits for Different Quantum Computers. In: The 1st International Workshop on Quantum Software Analysis, Evolution and Reengineering (Q-SANER 2022) co-located with IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 2022.

Approach – NISQ Analyzer



Backend Name	Provider	SDK	Width	Depth	Multi-Qubit Gate Depth	Total Number of Operations
ibmq_qasm_simulator	ibmq	pytket	3	14	6	21
ibmq_qasm_simulator	ibmq	qiskit	3	14	6	21
ibmq_bogota	ibmq	pytket	3	27	9	45
ibmq_bogota	ibmq	qiskit	3	30	13	51
ibmq_lima	ibmq	pytket	3	27	9	45
ibmq_lima	ibmq	qiskit	3	40	11	62
ibmq_belem	ibmq	pytket	3	29	9	45
ibmq_belem	ibmq	qiskit	3	41	11	63
ibmq_quito	ibmq	pytket	3	27	9	45

[1] Marie Salm, Johanna Barzen, Frank Leymann, Benjamin Weder, Karoline Wild. 2021. Automating the Comparison of Quantum Compilers for Quantum Circuits. In: Proceedings of the 15th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2021), Springer International Publishing, 2021

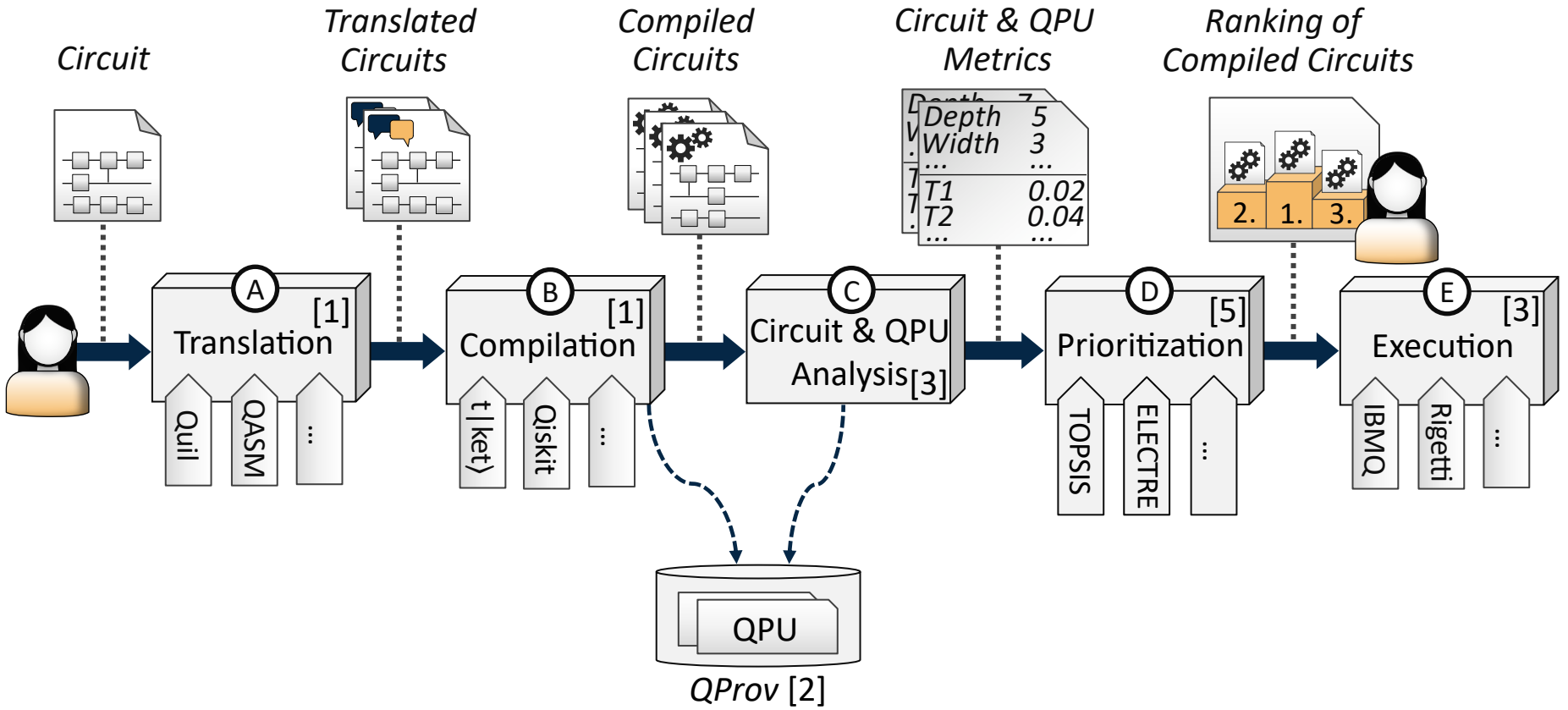
[2] Benjamin Weder, Johanna Barzen, Frank Leymann, Marie Salm, Karoline Wild. 2021. QProv: A provenance system for quantum computing. In: IET Quantum Communications, Wiley, 2021

[3] Marie Salm, Johanna Barzen, Uwe Breitenbücher, Frank Leymann, Benjamin Weder, et al. 2020. The NISQ Analyzer: Automating the Selection of Quantum Computers for Quantum Algorithms. In: Proceedings of the 14th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2020), Springer International Publishing, 2020

[4] Eyob A. Sete, William J. Zeng, and Chad T. Rigetti. 2016. A functional architecture for scalable quantum computing. In 2016 IEEE International Conference on Rebooting Computing(ICRC).1–6.

[5] Marie Salm, Johanna Barzen, Frank Leymann, Benjamin Weder: Prioritization of Compiled Quantum Circuits for Different Quantum Computers. In: The 1st International Workshop on Quantum Software Analysis, Evolution and Reengineering (Q-SANER 2022) co-located with IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 2022.

Approach – NISQ Analyzer



[1] Marie Salm, Johanna Barzen, Frank Leymann, Benjamin Weder, Karoline Wild. 2021. Automating the Comparison of Quantum Compilers for Quantum Circuits. In: Proceedings of the 15th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2021), Springer International Publishing, 2021

[2] Benjamin Weder, Johanna Barzen, Frank Leymann, Marie Salm, Karoline Wild. 2021. QProv: A provenance system for quantum computing. In: IET Quantum Communications, Wiley, 2021

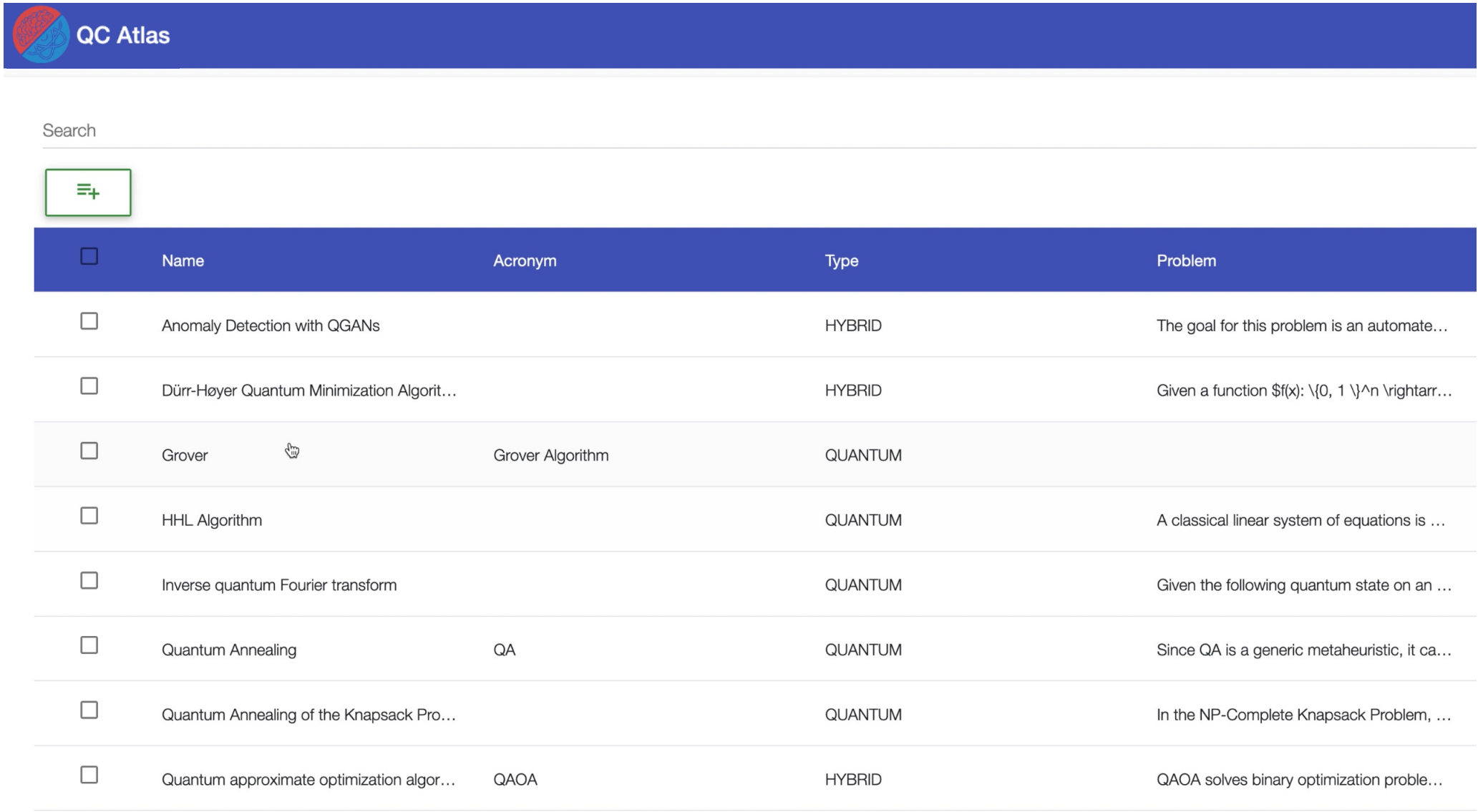
[3] Marie Salm, Johanna Barzen, Uwe Breitenbücher, Frank Leymann, Benjamin Weder, et al. 2020. The NISQ Analyzer: Automating the Selection of Quantum Computers for Quantum Algorithms. In: Proceedings of the 14th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2020), Springer International Publishing, 2020

[4] Eyob A. Sete, William J. Zeng, and Chad T. Rigetti. 2016. A functional architecture for scalable quantum computing. In 2016 IEEE International Conference on Rebooting Computing(ICRC).1–6.

[5] Marie Salm, Johanna Barzen, Frank Leymann, Benjamin Weder: Prioritization of Compiled Quantum Circuits for Different Quantum Computers. In: The 1st International Workshop on Quantum Software Analysis, Evolution and Reengineering (Q-SANER 2022) co-located with IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), 2022.

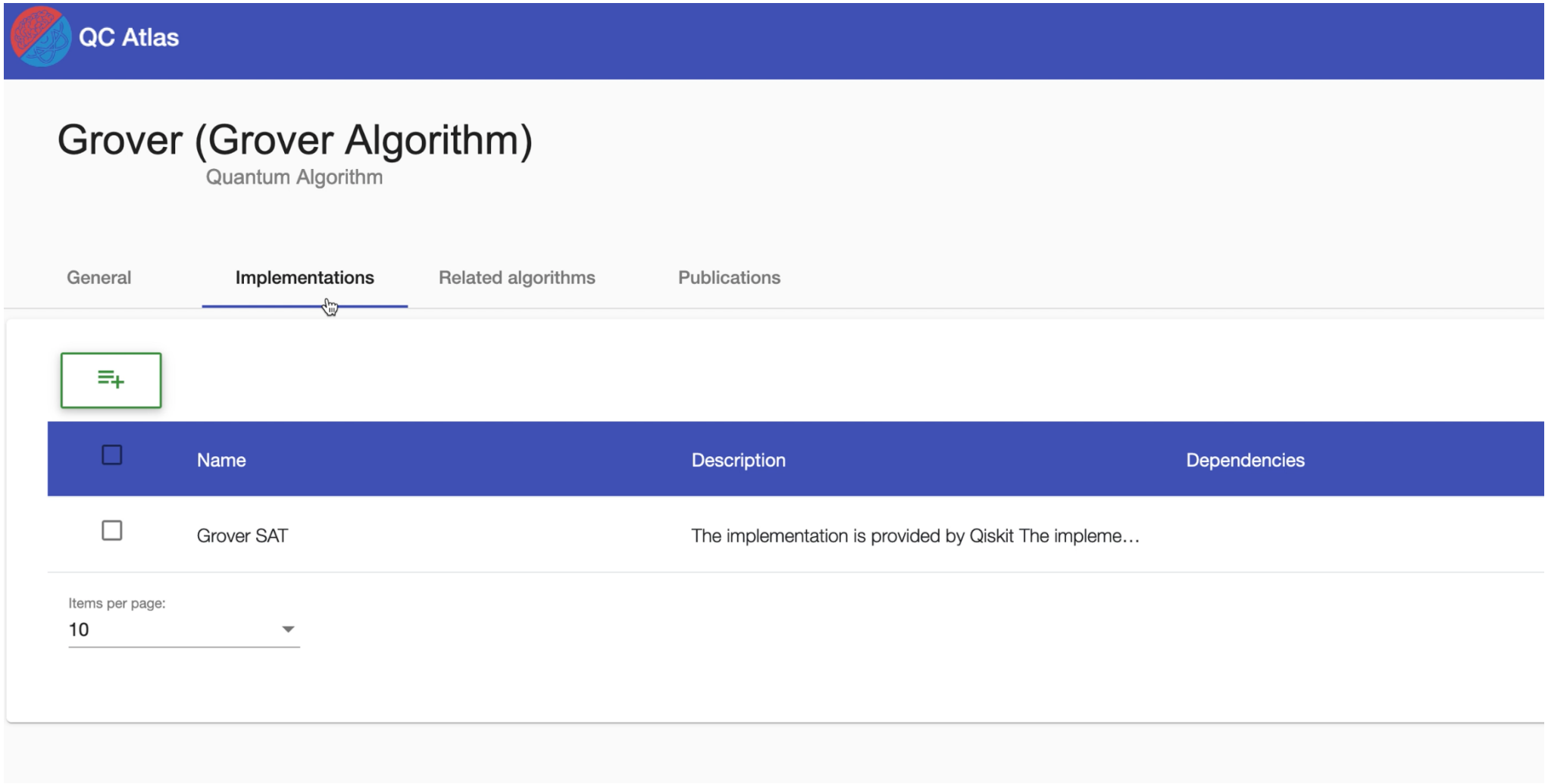
Prototype

List of algorithms



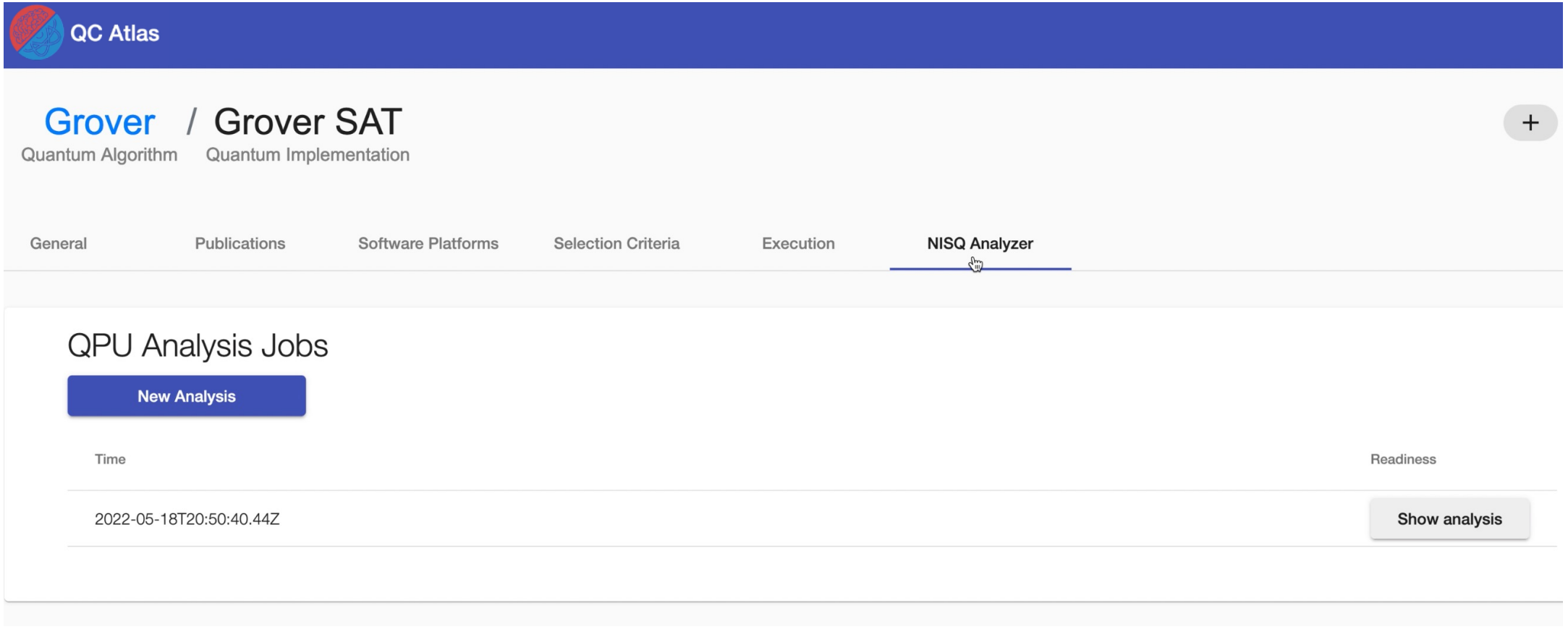
https://github.com/UST-QuAntiL/quantil-docker
https://quantil.readthedocs.io/en/latest/

Implementation of an algorithm



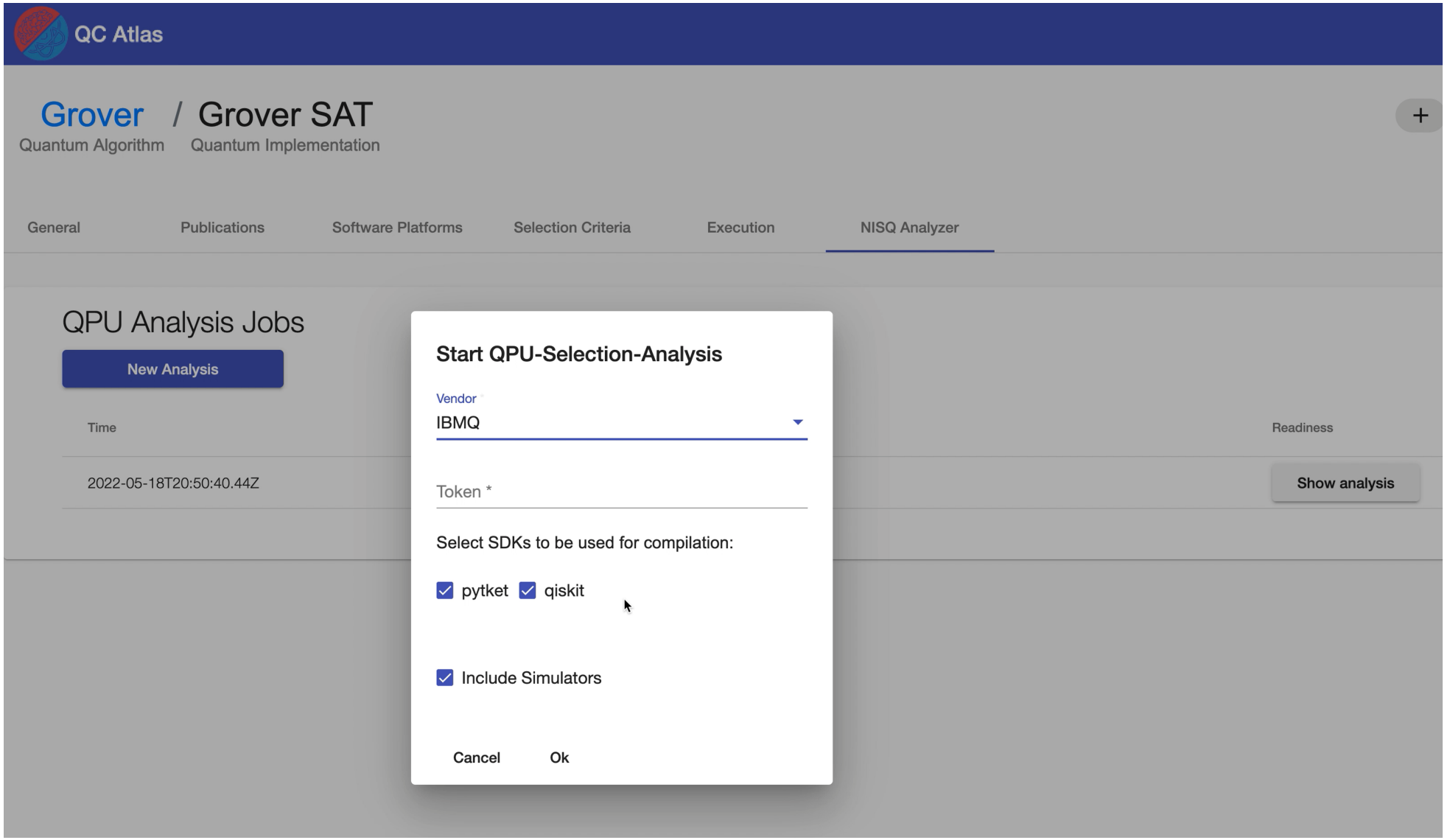
https://github.com/UST-QuAntiL/quantil-docker
https://quantil.readthedocs.io/en/latest/

NISQ Analyzer



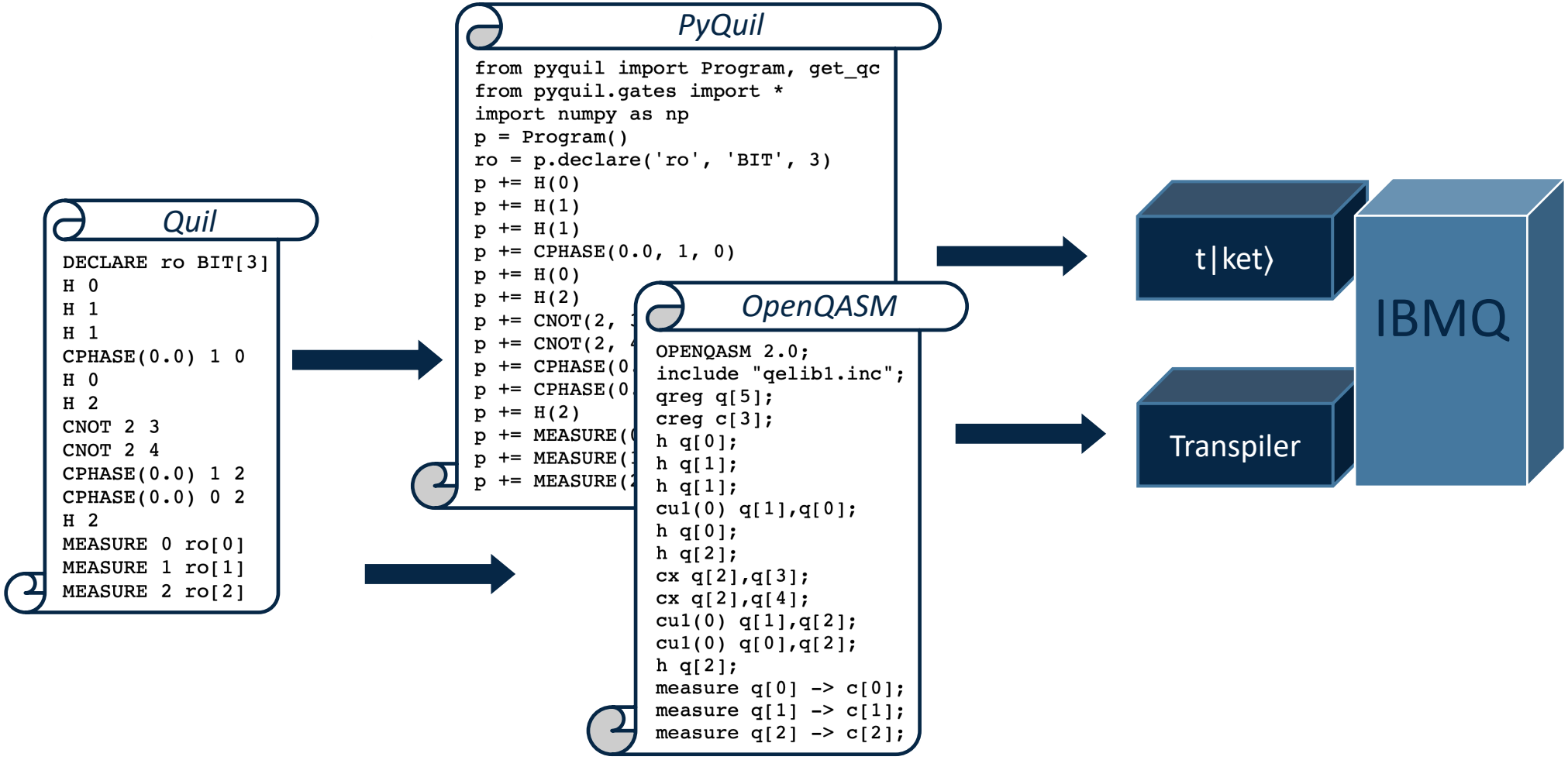
https://github.com/UST-QuAntiL/quantil-docker
https://quantil.readthedocs.io/en/latest/

Start analysis

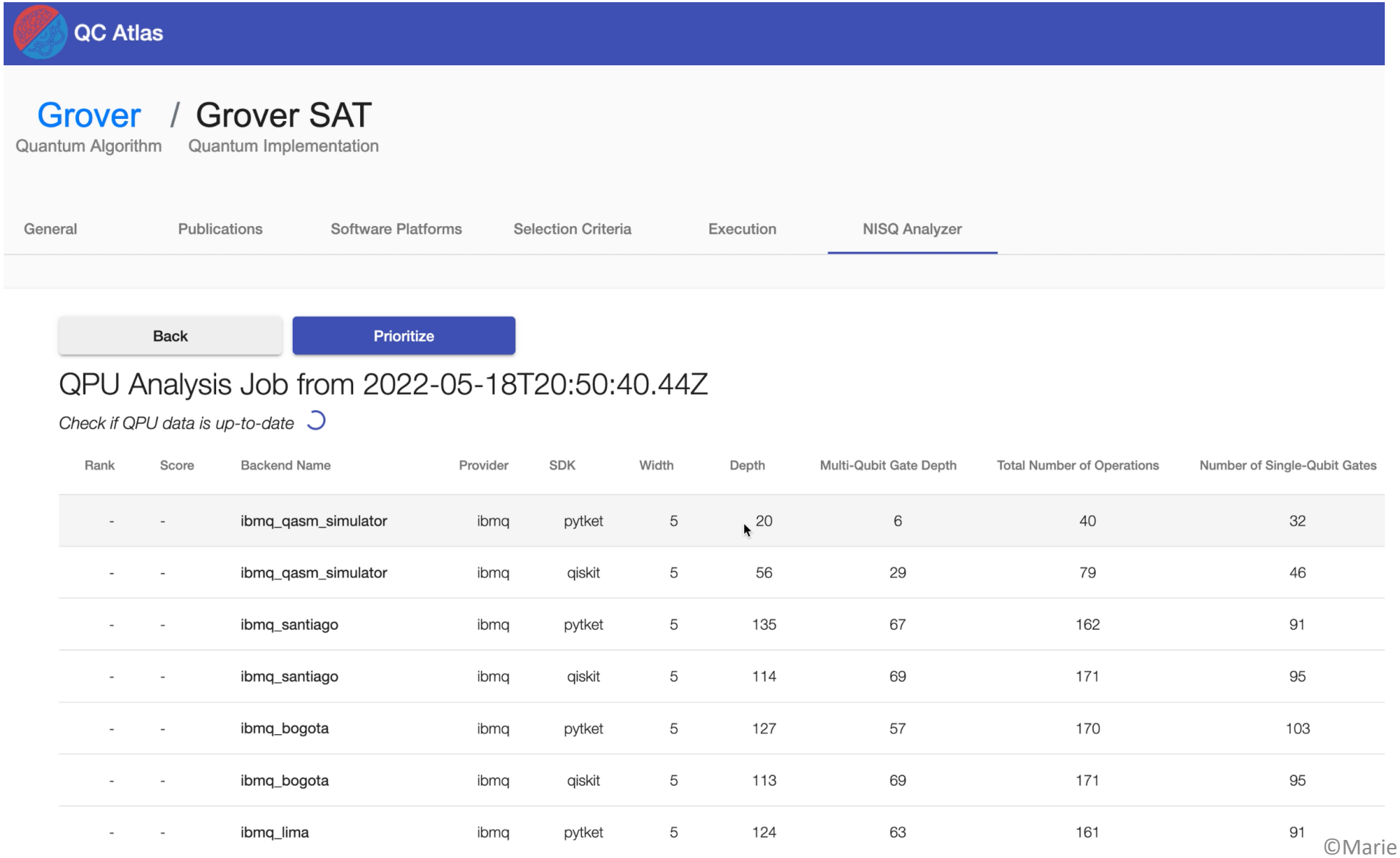


https://github.com/UST-QuAntiL/quantil-docker
https://quantil.readthedocs.io/en/latest/

Implementation in Quil for IBMQ

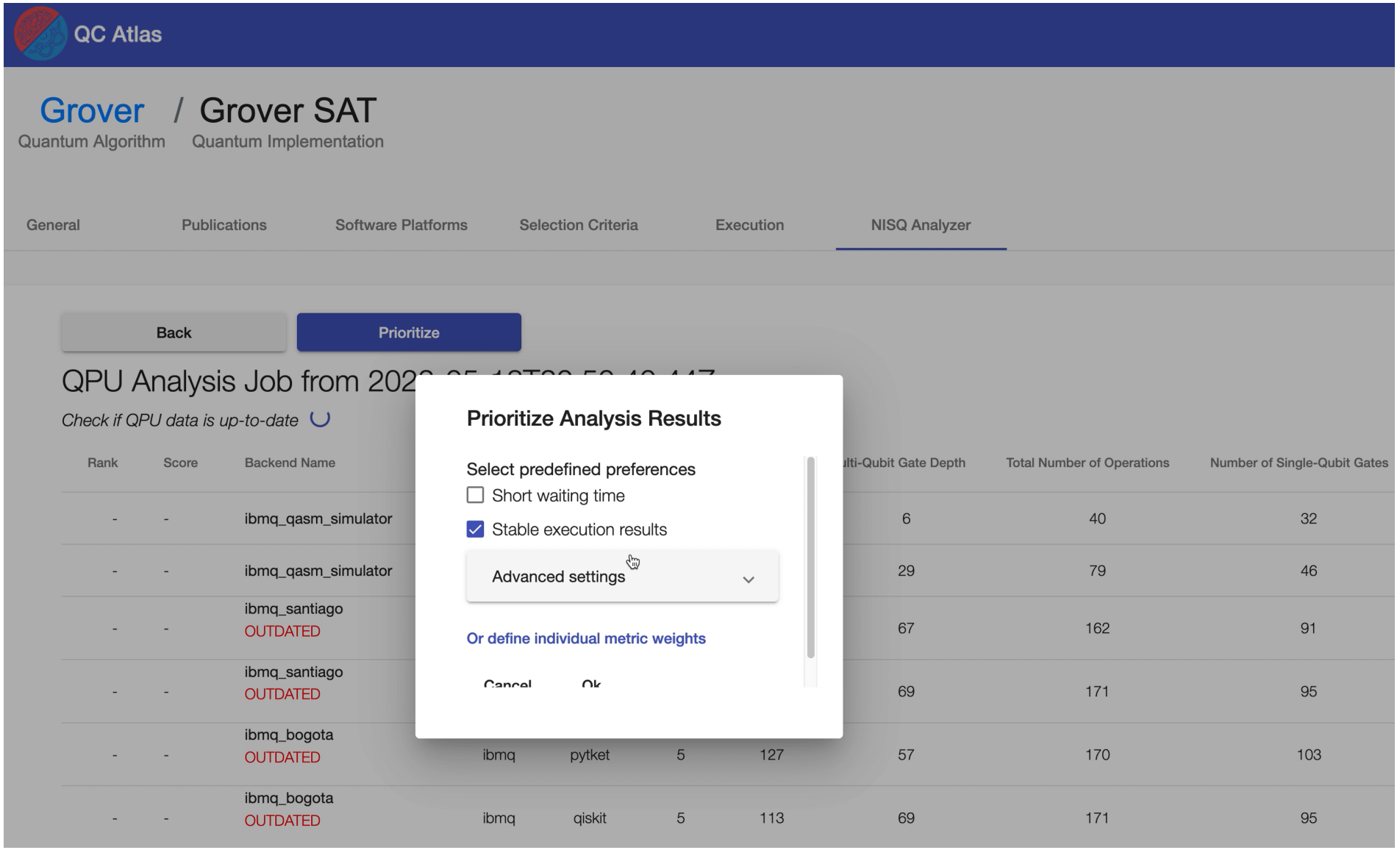


Compilation results

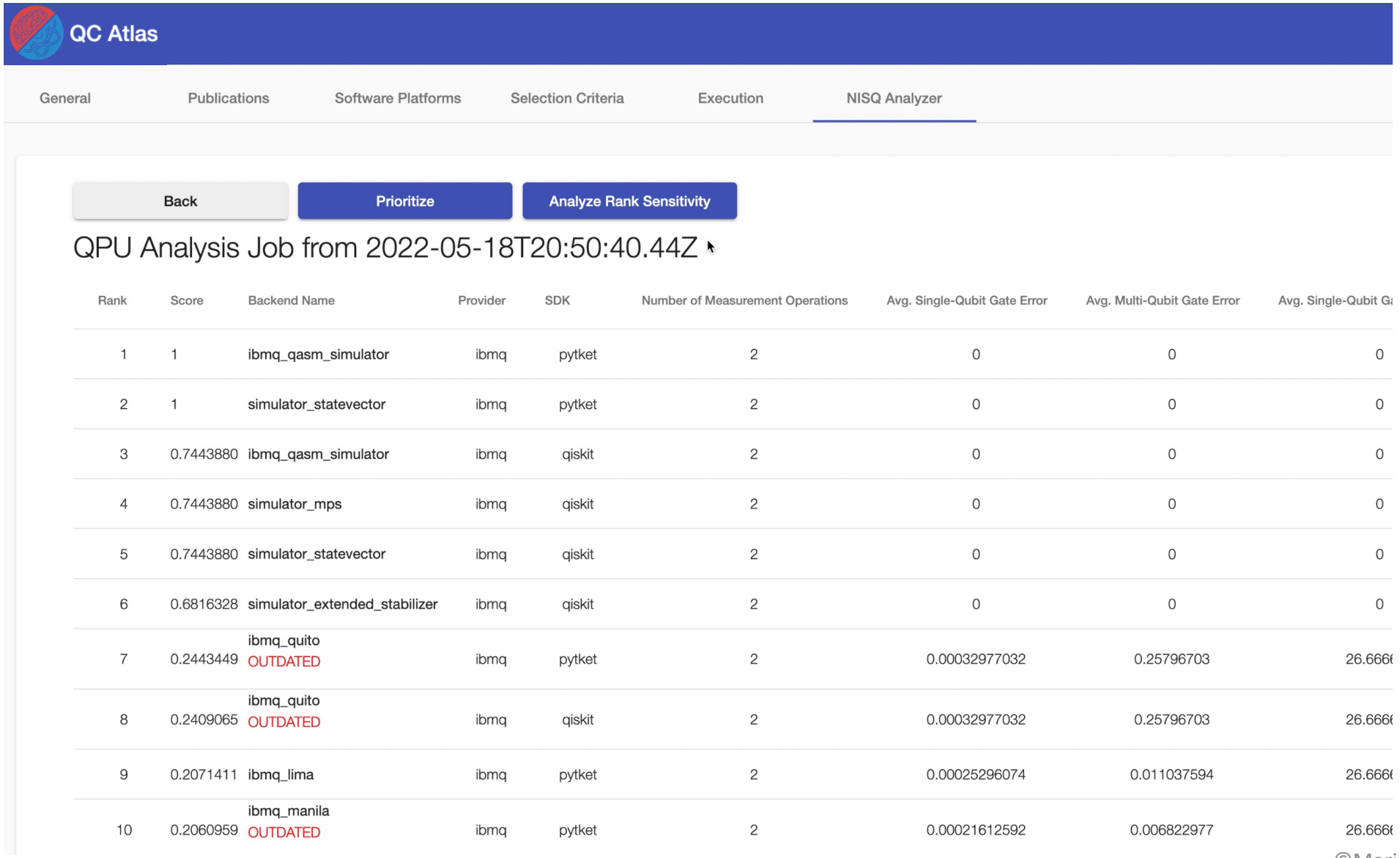


Rank	Score	Backend Name	Provider	SDK	Width	Depth	Multi-Qubit Gate Depth	Total Number of Operations	Number of Single-Qubit Gates
-	-	ibmq_qasm_simulator	ibmq	pytket	5	20	6	40	32
-	-	ibmq_qasm_simulator	ibmq	qiskit	5	56	29	79	46
-	-	ibmq_santiago	ibmq	pytket	5	135	67	162	91
-	-	ibmq_santiago	ibmq	qiskit	5	114	69	171	95
-	-	ibmq_bogota	ibmq	pytket	5	127	57	170	103
-	-	ibmq_bogota	ibmq	qiskit	5	113	69	171	95
-	-	ibmq_lima	ibmq	pytket	5	124	63	161	91

Prioritization of compiled circuits

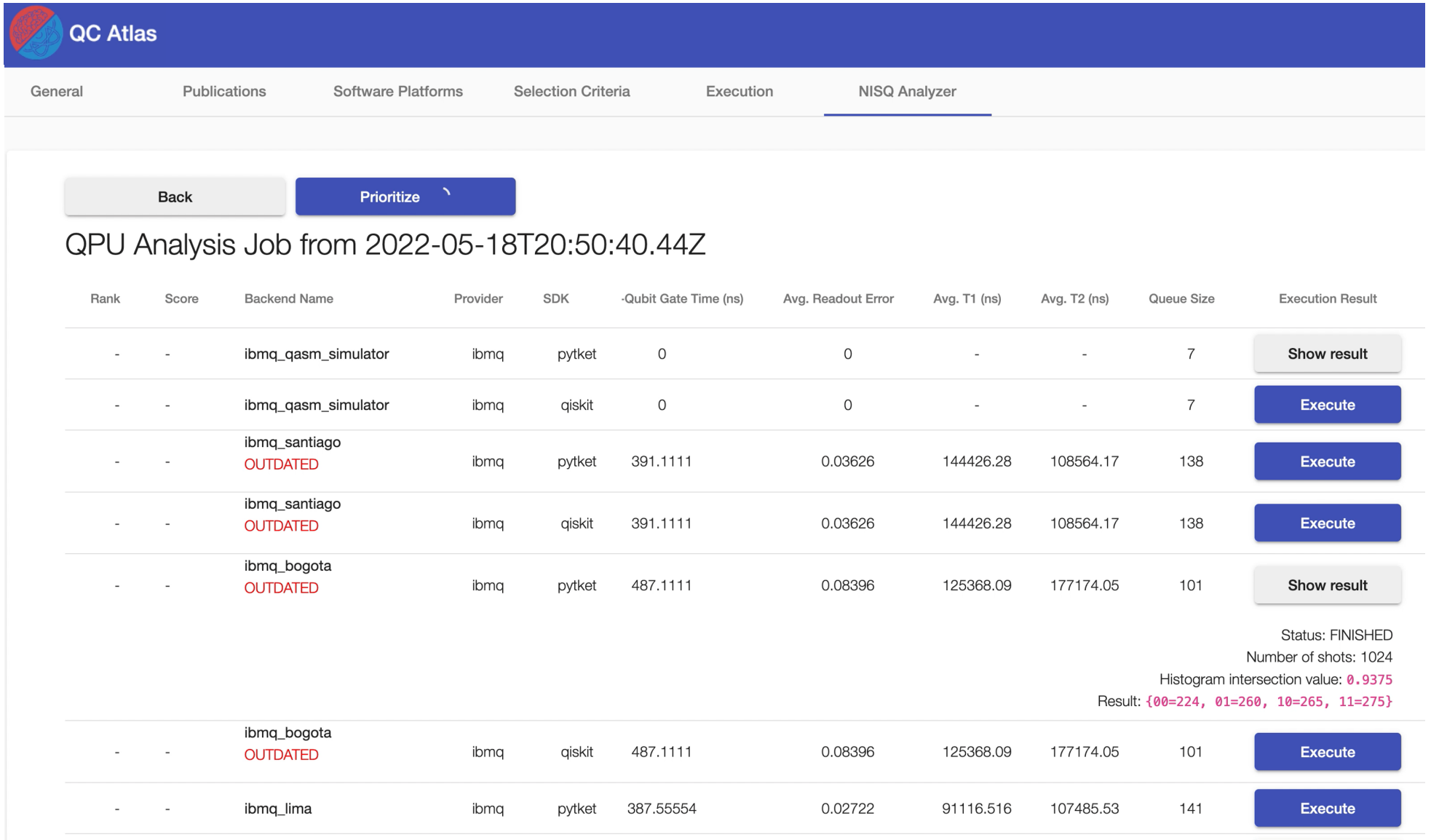


Ranked list of compilation results



Rank	Score	Backend Name	Provider	SDK	Number of Measurement Operations	Avg. Single-Qubit Gate Error	Avg. Multi-Qubit Gate Error	Avg. Single-Qubit Ga
1	1	ibmq_qasm_simulator	ibmq	pytket	2	0	0	0
2	1	simulator_statevector	ibmq	pytket	2	0	0	0
3	0.7443880	ibmq_qasm_simulator	ibmq	qiskit	2	0	0	0
4	0.7443880	simulator_mps	ibmq	qiskit	2	0	0	0
5	0.7443880	simulator_statevector	ibmq	qiskit	2	0	0	0
6	0.6816328	simulator_extended_stabilizer	ibmq	qiskit	2	0	0	0
7	0.2443449	ibmq_quito OUTDATED	ibmq	pytket	2	0.00032977032	0.25796703	26.666
8	0.2409065	ibmq_quito OUTDATED	ibmq	qiskit	2	0.00032977032	0.25796703	26.666
9	0.2071411	ibmq_lima	ibmq	pytket	2	0.00025296074	0.011037594	26.666
10	0.2060959	ibmq_manila OUTDATED	ibmq	pytket	2	0.00021612592	0.006822977	26.666

Execution


QC Atlas
General
Publications
Software Platforms
Selection Criteria
Execution
NISQ Analyzer
Back
Prioritize
QPU Analysis Job from 2022-05-18T20:50:40.44Z
Rank
Score
Backend Name
Provider
SDK
-Qubit Gate Time (ns)
Avg. Readout Error
Avg. T1 (ns)
Avg. T2 (ns)
Queue Size
Execution Result
- - ibmq_qasm_simulator ibmq pytket 0 0 - - 7 Show result
- - ibmq_qasm_simulator ibmq qiskit 0 0 - - 7 Execute
- - ibmq_santiago OUTDATED ibmq pytket 391.1111 0.03626 144426.28 108564.17 138 Execute
- - ibmq_santiago OUTDATED ibmq qiskit 391.1111 0.03626 144426.28 108564.17 138 Execute
- - ibmq_bogota OUTDATED ibmq pytket 487.1111 0.08396 125368.09 177174.05 101 Show result
Status: FINISHED
Number of shots: 1024
Histogram intersection value: 0.9375
Result: {00=224, 01=260, 10=265, 11=275}
- - ibmq_bogota OUTDATED ibmq qiskit 487.1111 0.08396 125368.09 177174.05 101 Execute
- - ibmq_lima ibmq pytket 387.55554 0.02722 91116.516 107485.53 141 Execute

Input-sensitive implementation selection

How to find implementations that can execute the input data?

Factorizing

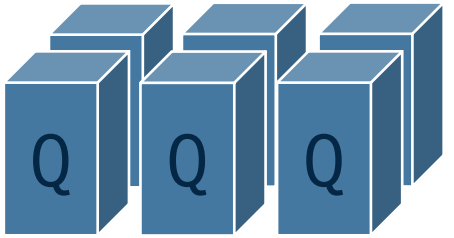
N = 255

Shor!

255 *ok?*

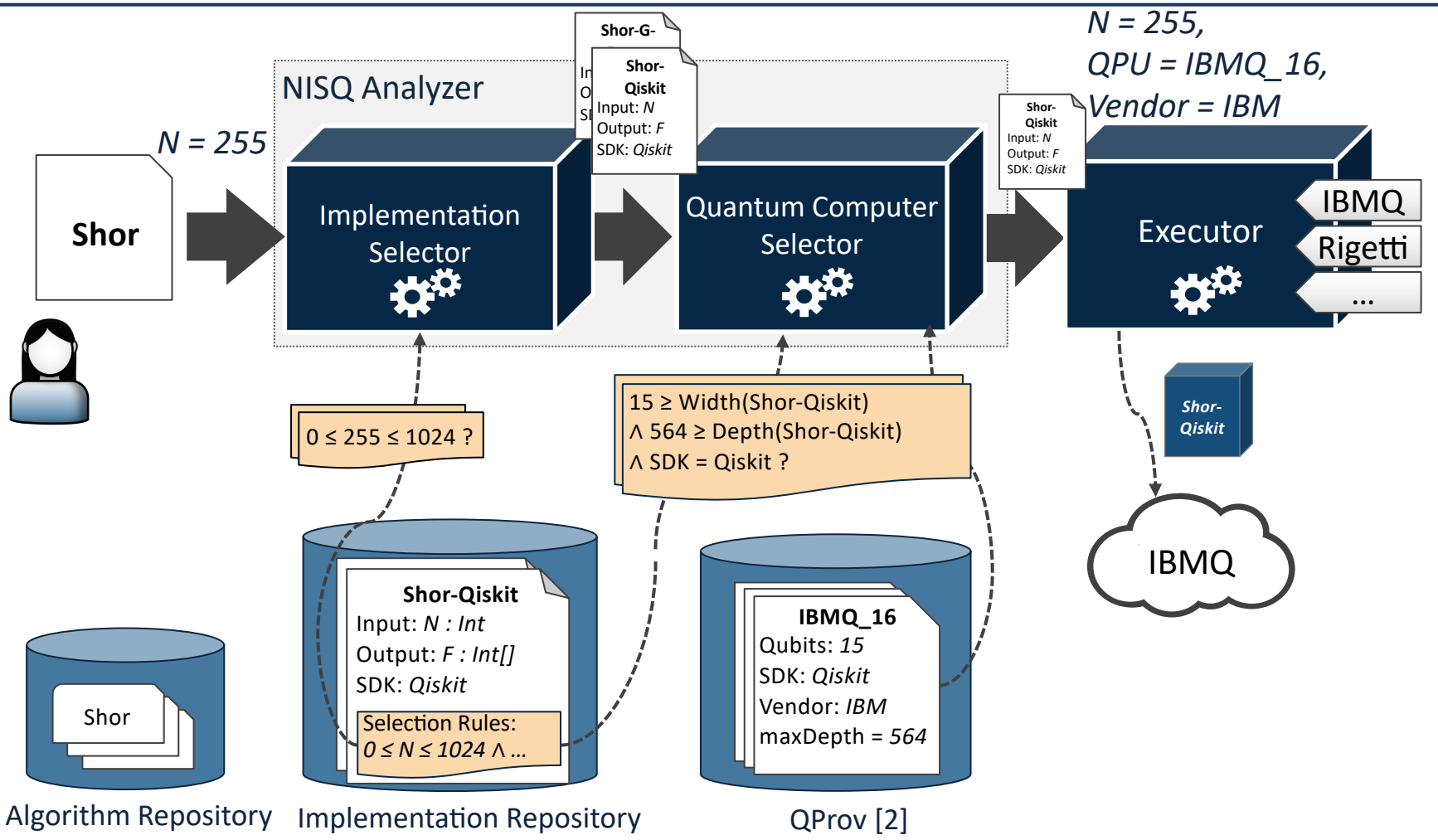
+

enough qubits, coherence time, ...?



immense effort & knowledge needed!

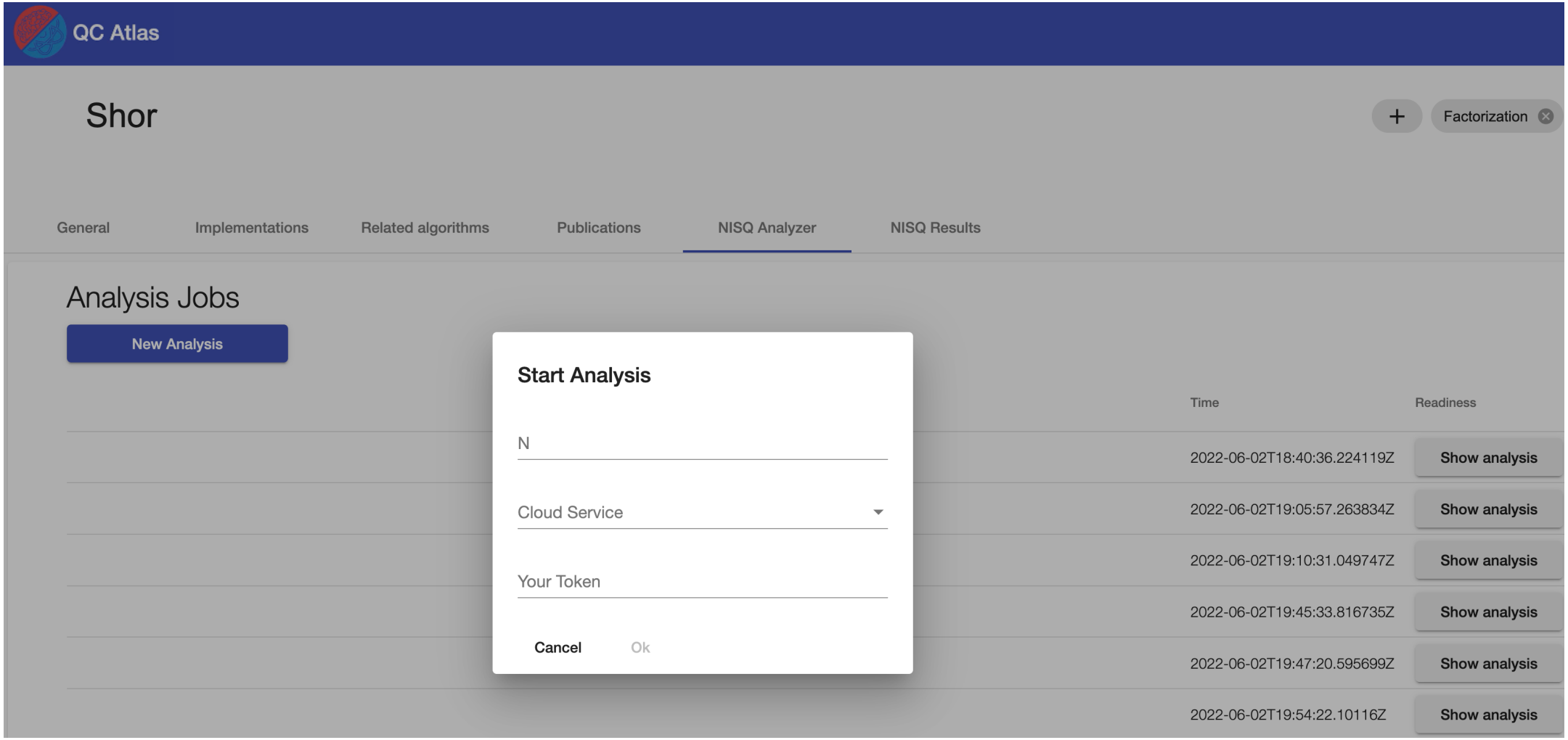
NISQ Analyzer – Implementation & QPU Selection [1]



[1] Marie Salm, Johanna Barzen, Uwe Breitenbücher, Frank Leymann, Benjamin Weder, et al. 2020. The NISQ Analyzer: Automating the Selection of Quantum Computers for Quantum Algorithms. In: Proceedings of the 14th Symposium and Summer School on Service-Oriented Computing (SummerSOC 2020), Springer International Publishing, 2020

[2] Weder, Benjamin; Barzen, Johanna; Leymann, Frank; Salm, Marie; Wild, Karoline: QProv: A Provenance System for Quantum Computing. In: IET Quantum Communications, Wiley, 2021

Analysis based on input data



https://github.com/UST-QuAntiL/quantil-docker
https://quantil.readthedocs.io/en/latest/

Compilation results of different implementations

QC Atlas

Check if QPU data is up-to-date

Implementation: **shor-fix**

Backend Name	Provider	SDK	Width	Depth	Multi-Qubit Gate Depth	Total Number of Operations	Number of Single-Qubit Gates	Number of Multi-Qubit Gates
ibmq_qasm_simulator	ibmq	qiskit	5	7	4	10	2	5
ibmq_santiago	ibmq	qiskit	5	15	3	24	15	6
ibmq_bogota	ibmq	qiskit	5	17	8	26	12	11
ibmq_lima	ibmq	qiskit	4	12	5	14	6	5
ibmq_belem	ibmq	qiskit	5	12	7	17	6	8
ibmq_quito	ibmq	qiskit	5	12	7	17	6	8
simulator_statevector	ibmq	qiskit	5	7	4	10	2	5
simulator_mps	ibmq	qiskit	5	6	4	9	1	5
simulator_extended_stabilizer	ibmq	qiskit	3	7	2	9	4	2
ibmq_manila	ibmq	qiskit	5	11	5	16	5	8

Implementation: **shor-general**

Backend Name	Provider	SDK	Width	Depth	Multi-Qubit Gate Depth	Total Number of Operations	Number of Single-Qubit Gates	Number of Multi-Qubit Gates
ibmq_qasm_simulator	ibmq	qiskit	5	7	4	10	2	5
ibmq_santiago	ibmq	qiskit	4	18	6	25	16	6
ibmq_lima	ibmq	qiskit	4	12	5	14	6	5

Conclusion

Conclusion

- to select/design & execute a circuit
  - QPU, compiler, SDK & access must be considered
  - for precise & fast executions

➢ automated selection, translation, compilation, prioritization & execution
  - plug-in based, supports extensibility

- in the future
  - further SDKs
  - deeper analyzes
  - integration into QHAna

Thank you!