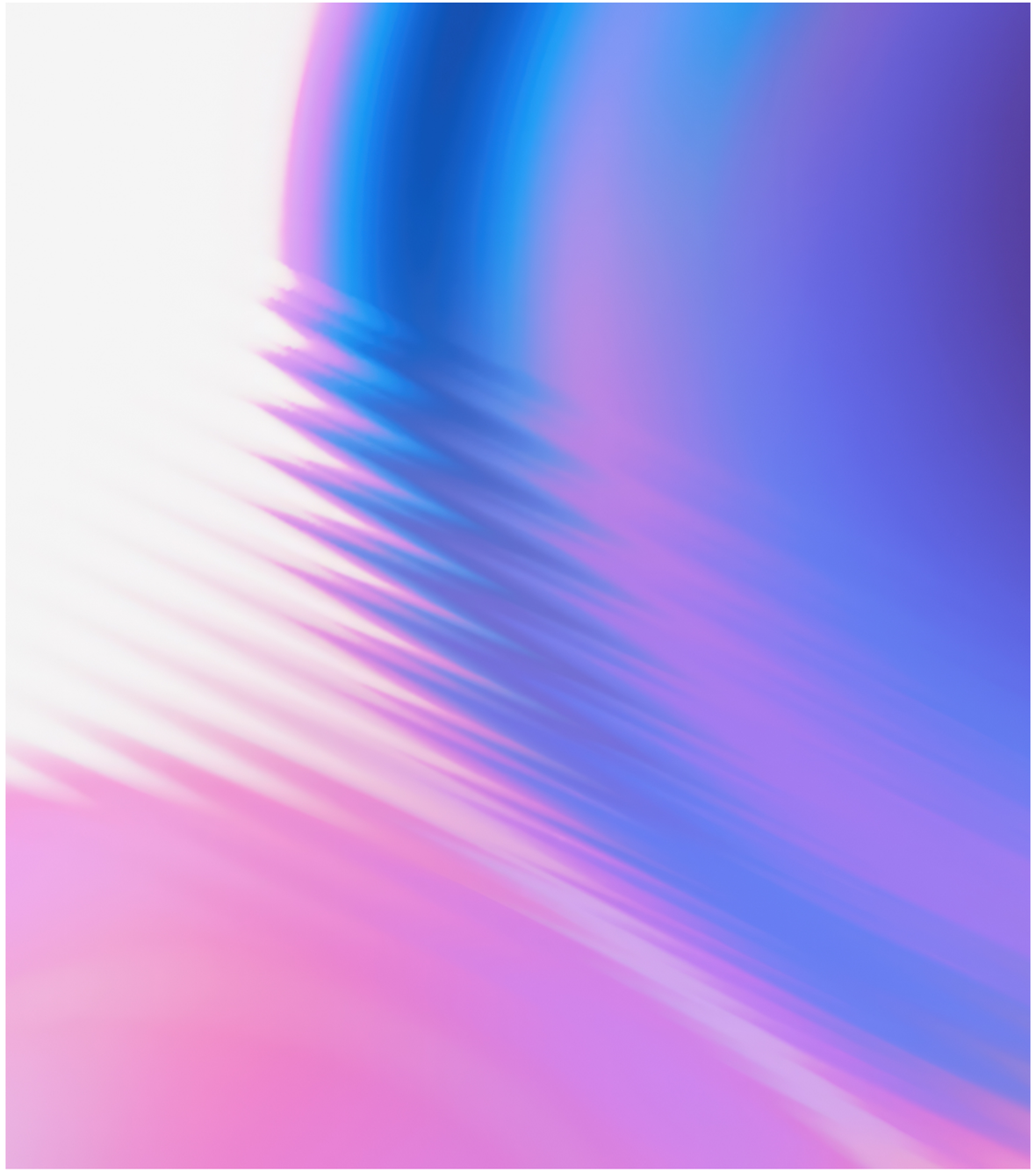
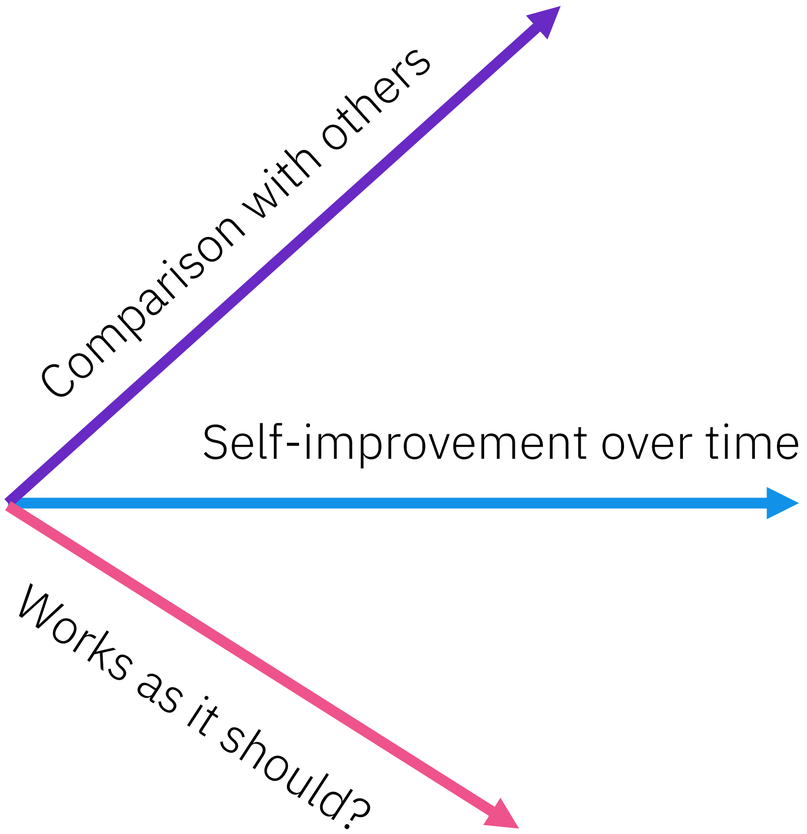
Benchmarking the performance of quantum computing software

Paul Nation
Principal Research Scientist
IBM Quantum

3rd International TQCI Seminar
June 24-25th , 2025



Benchmarking in a nutshell



Benchmarking is a standardized comparison based on metrics such as time, quality, and other measures

- Comparisons against peers for competitive analysis / positioning

- Self-comparisons tracking movement against a set of metrics aligned with objectives

- Does it do what it should?

Provides a quantitative means by which progress (or degradation) can be gauged

Why focus on benchmarking now?

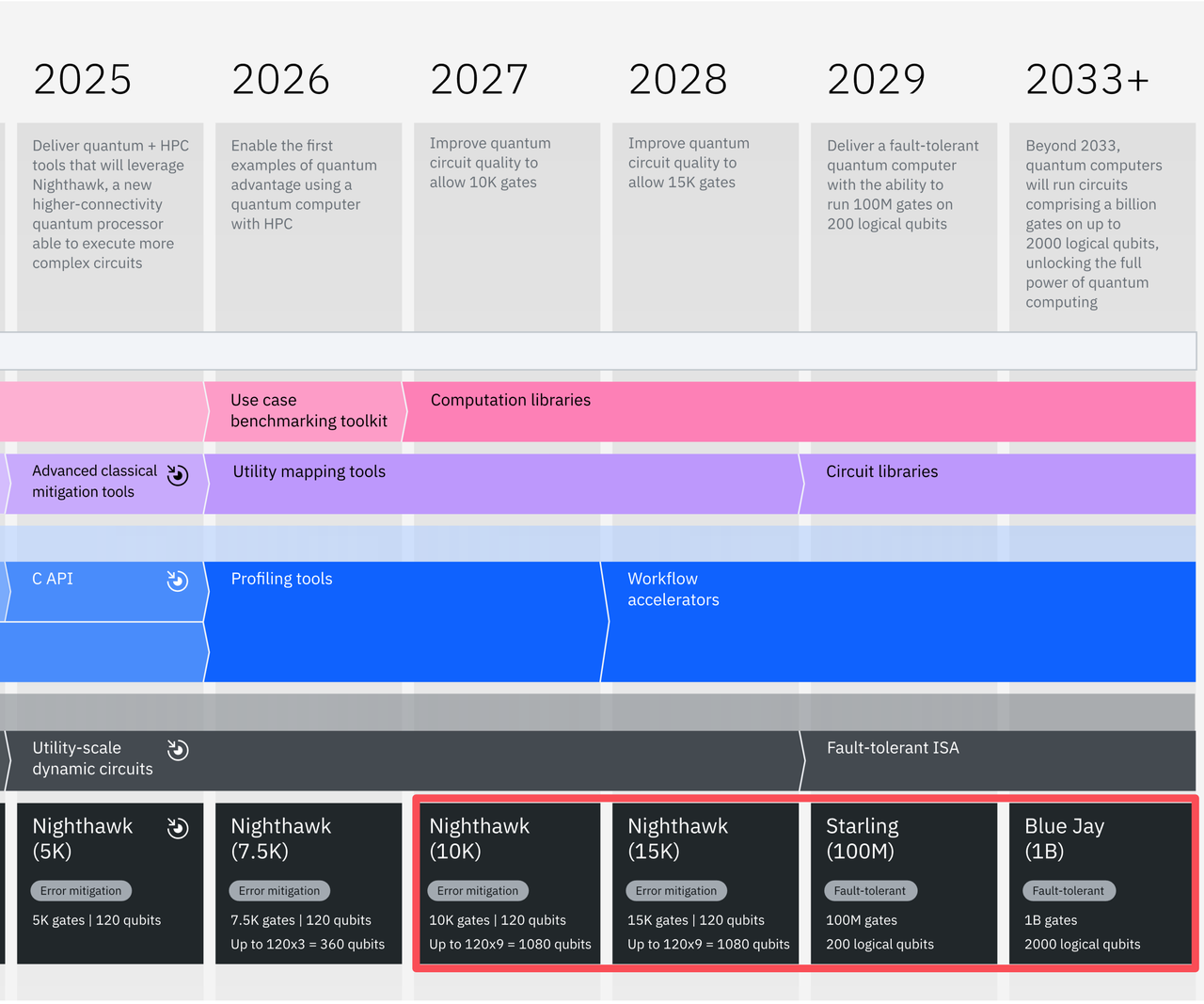
The quantum computing landscape has changed substantially in the last few years.

- Multiple HW vendors accessible via the cloud
- Governments purchasing multiple hardware modalities for evaluation
- Wide variety of software frameworks at level of quantum circuits
- Established computing firms looking to capture mindshare

There are hardware and software performance claims almost daily

Never assume results hold true; always be data driven through open and transparent benchmarks

Near-term circuit scaling



Roadmaps show quantum hardware capable of executing **millions of gates over thousands of qubits** within 5 years

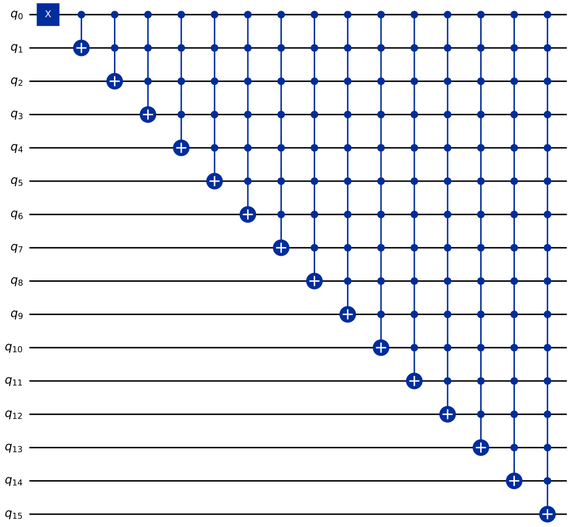
Nearly all quantum circuits need preprocessing by quantum software before execution

Quantum software needs to remain performant at these scales (and beyond) not to be a bottleneck

As algorithm and application complexity increases, the impact of quantum software on the results grows in tandem

Software impact on abstract circuits

Multi-controlled circuit



Synthesize to Rx, Ry, Rz, CZ

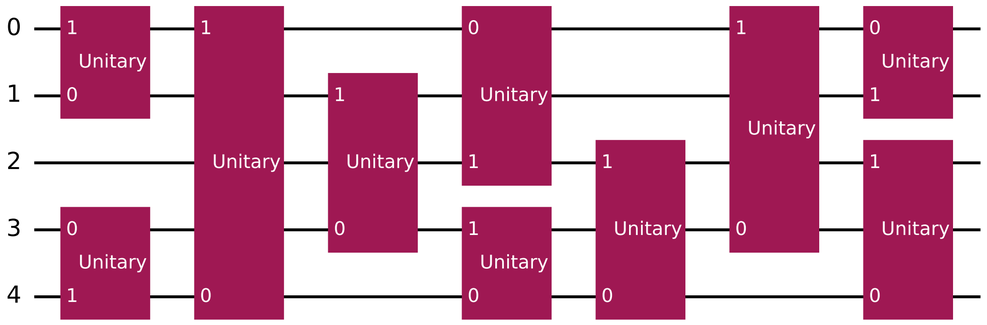
BQSKit = Failed

Cirq = 17, 414 CZ

Qiskit = 2,725 CZ

Tket = 4,457 CZ

Quantum Volume 100 (square)



Synthesize + mapping
+ gate optimization

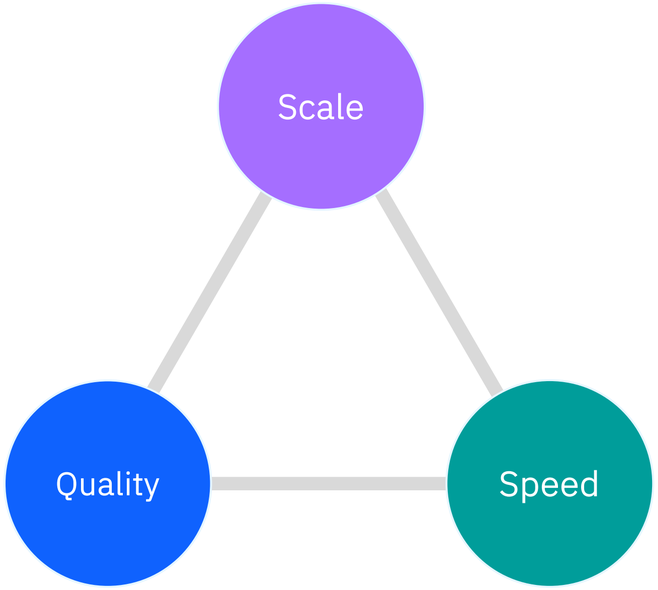
BQSKit = 82,345 CZ / depth = 12,094

Qiskit = 76,608 CZ / depth = 8,988

Tket = 76,203 CZ / depth = 8,523

BQSkit 1.2.0, Cirq 1.5.0, Qiskit 2.0.2, Tket 2.6.0

Benchmarking quantum software

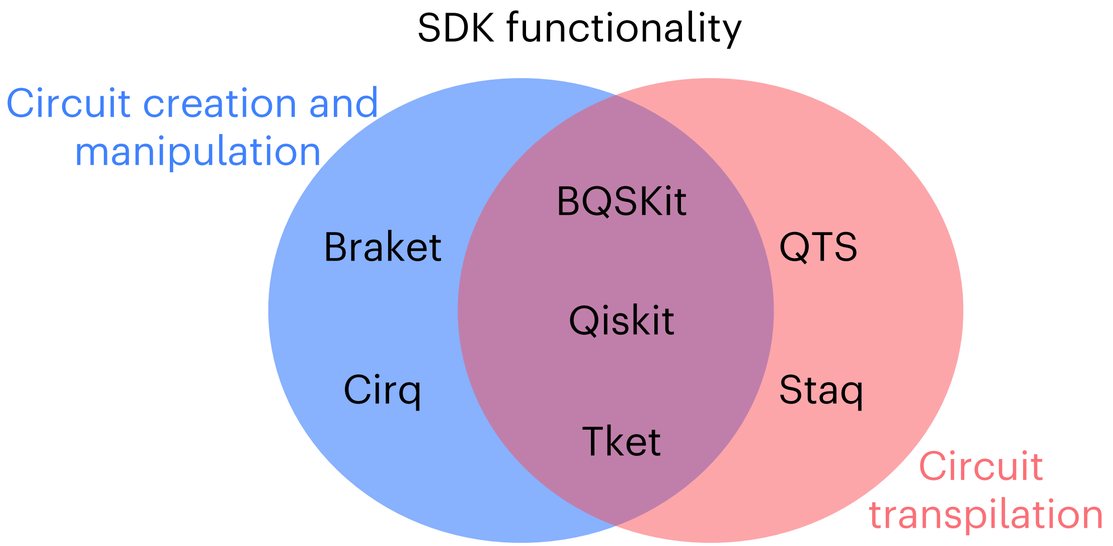


Scale: Software performance as a function of qubit count / number of operations

Quality: Output circuit properties such as 2Q gate count, 2Q gate depth, T-gate count, etc.

Speed: Wall clock time to perform operations

Not all quantum software is alike; can loosely be partitioned into three categories



Benchpress

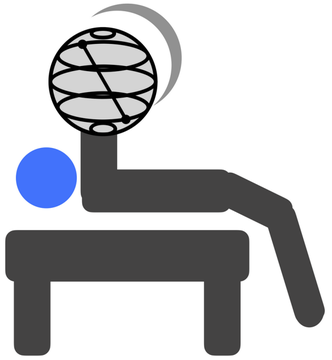
Benchmarking quantum software for quantum circuit creation, manipulation, and optimization

1000+ tests from ourselves, Feynman, QasmBench, and Hamlib libraries

Up to **930 qubits and $O(10^6)$ two-qubit gates**

Designed to probe a wide range of SDK functionality across software with disparate feature sets

Leverages Qiskit's wide-ranging compatibility to define tests over snapshots of real systems (IBM Torino), as well as abstract topologies (all-to-all, square, heavy-hex, linear)



Benchpress

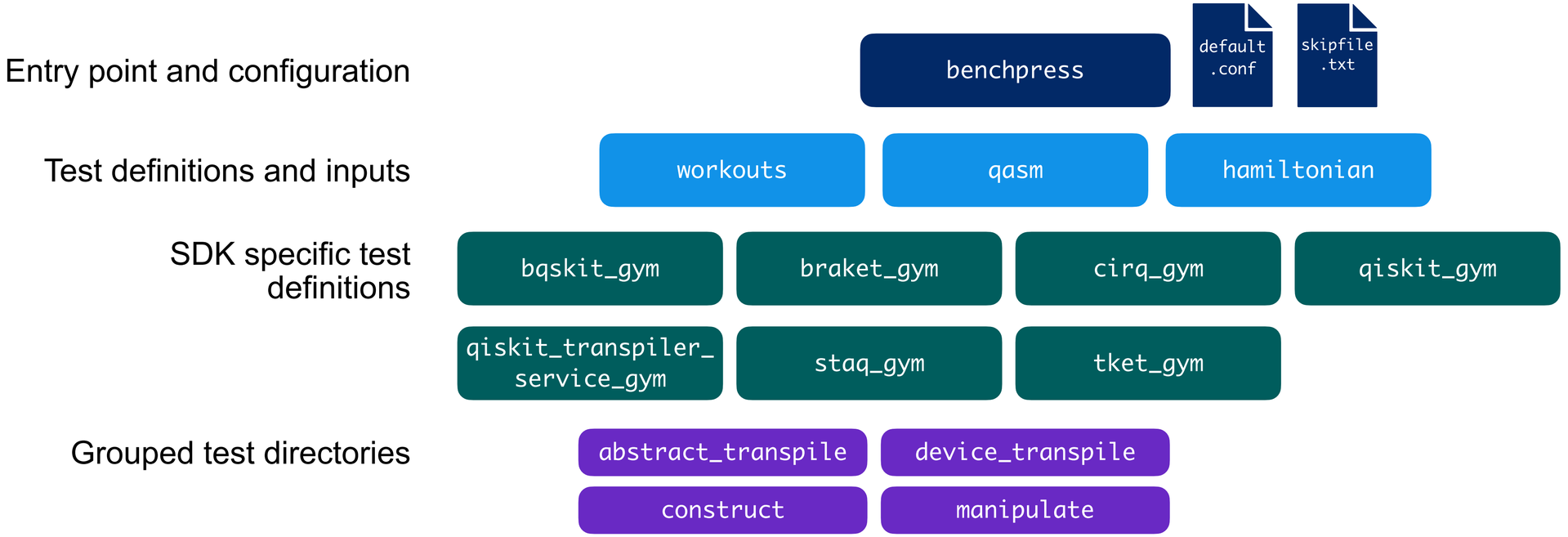
Repo: https://github.com/Qiskit/benchpress

Paper: Nat. Comput. Sci. 5, 427 (2025)

Tested SDKs:

- BQSKit (LBNL)
- Braket (AWS)
- Cirq (Google)
- Qiskit (IBM)
- Qiskit Transpiler Service (QTS)
- Staq (Waterloo)
- Tket (Quantinuum)

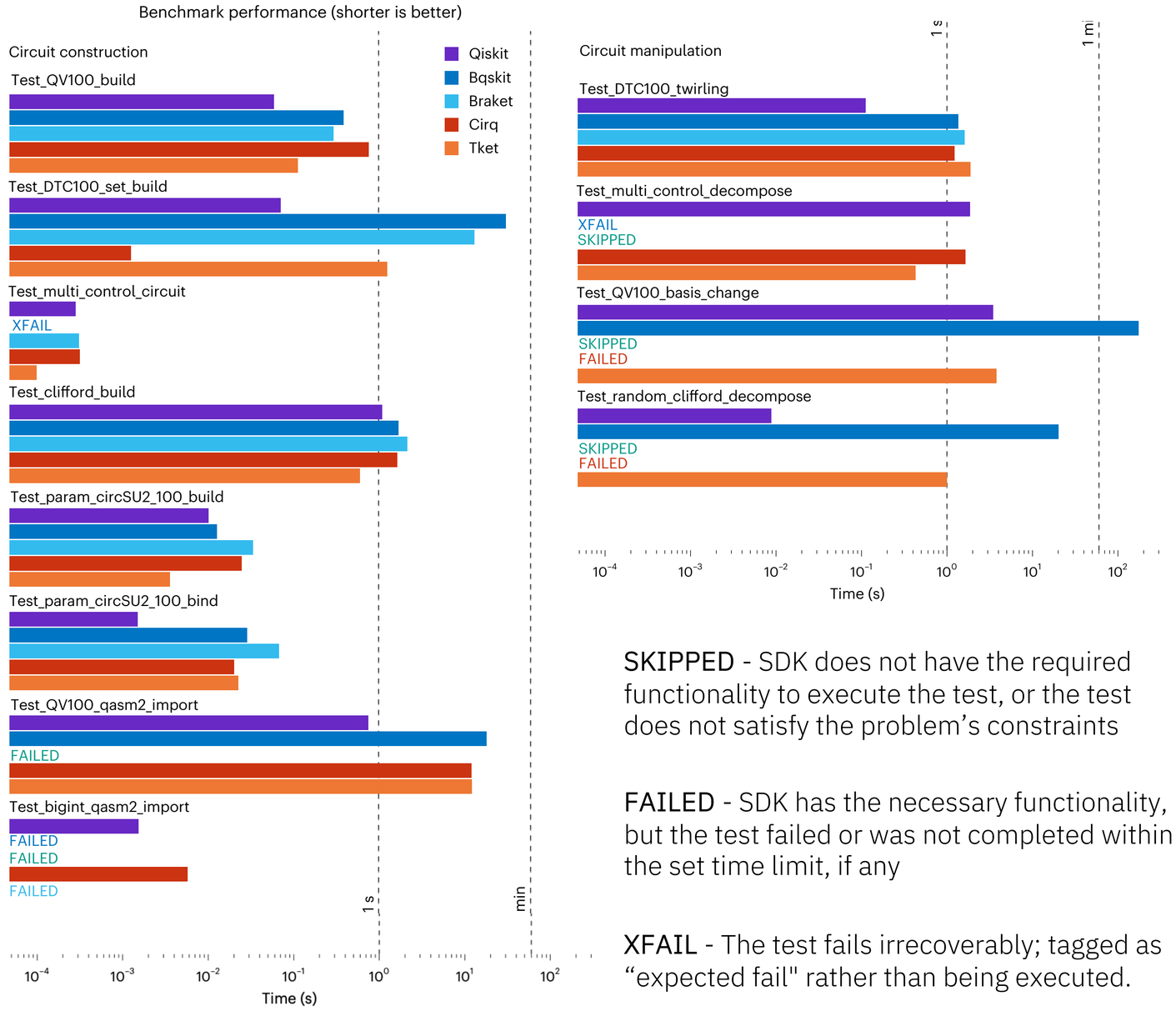
Benchpress



Benchpress defines a single suite of tests; all tests are processed regardless of if they are supported by the target SDK

Simple validations of inputs and outputs, and these have caught several hidden SDK issues

Benchpress: Circuit creation and manipulation



SKIPPED - SDK does not have the required functionality to execute the test, or the test does not satisfy the problem's constraints

FAILED - SDK has the necessary functionality, but the test failed or was not completed within the set time limit, if any

XFAIL - The test fails irrecoverably; tagged as "expected fail" rather than being executed.

Construction and manipulation of quantum circuits are core routines

Individually not a lot of time, but often need repeat calls

- e.g., binding parameters, gate insertion / removal, etc...

Tests need to be written in their own SDK language

Begin to see the disparate functionality between SDKs

Benchpress: Transpilation

Paper results:

Qiskit 1.2 vs Tket 1.31

24%

Mean reduction in 2Q-gate count

13x

Mean transpilation time improvement

Latest results:

Qiskit 2.0.2 vs Tket 2.6.0:

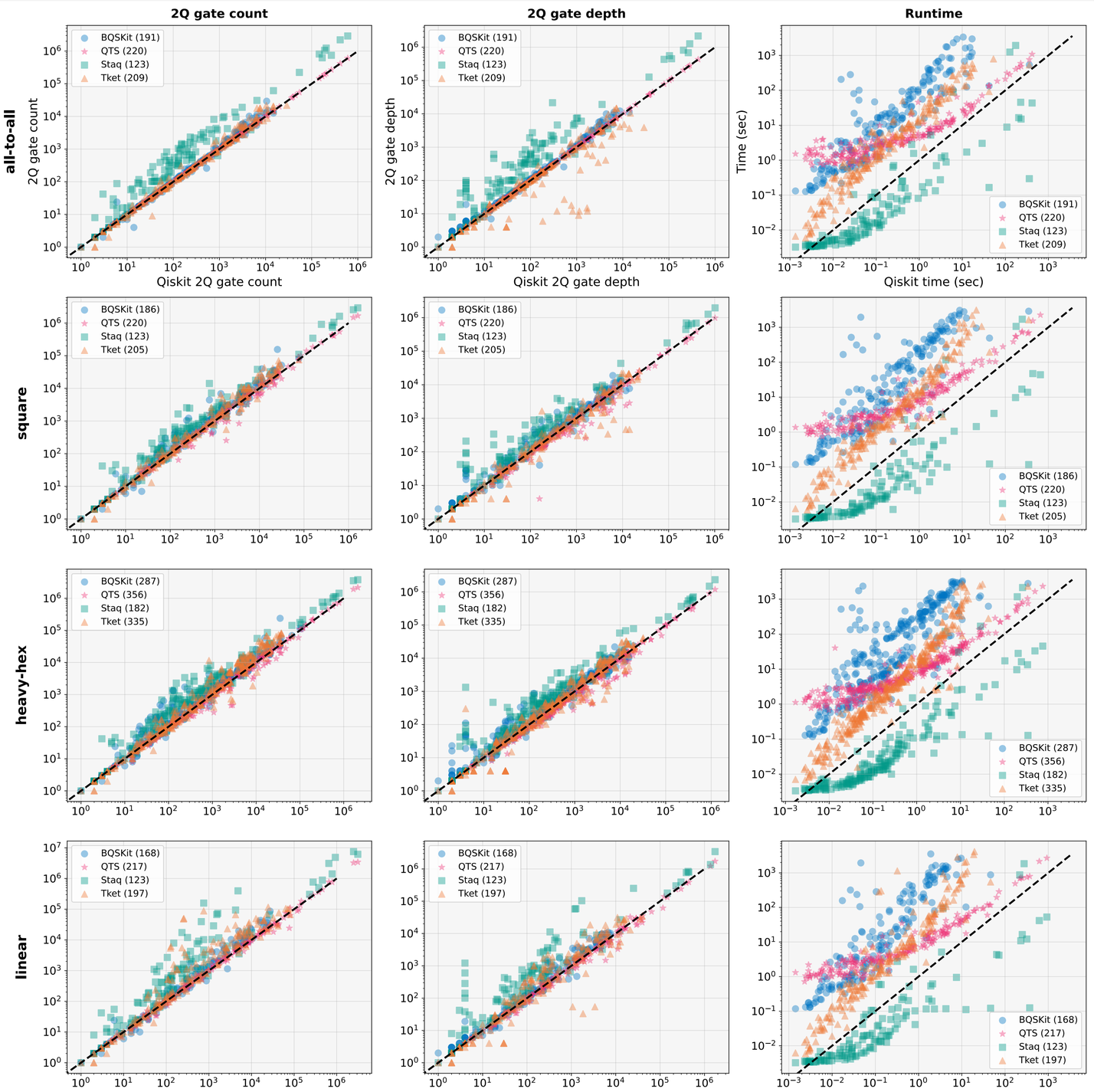
28%

Mean reduction in 2Q-gate count

63x

Mean transpilation time improvement

[Qiskit vs BQSKit: **18%** and **514x**]



Benchpress results

Qiskit is the only software to not fail a single test

	PASSED	SKIPPED	FAILED	XFAIL
BQSKit	841	22	201	2
Braket	7	1057	2	0
Cirq	10	1054	2	0
Qiskit	1044	22	0	0
QTS	1013	34	19	0
Staq	549	515	2	0
Tket	957	22	87	0

Results from paper

Vast majority of FAILED due to imposed timeout of 1 hour / test

Which tests pass or fail can vary across SDK version numbers

Tket passing tests only:

9min

Qiskit 2.0.2

19hrs

Tket 2.6.0

All tests:

33min

>65hrs

Can only give lower-bound on Tket time due to failed tests

Benchpress

Do not assume I am telling the truth; find out for yourself:

https://github.com/Qiskit/benchpress

- All tests and execution framework are open-source
- All raw data associated with published results available as JSON
- Open an issue or make a PR if you want to contribute to the process

Or take it and use it in your own work:

“It's quite easy to use, and we are benchmarking other SDKs with this collection.”

- Referee #1

https://github.com/CQCL/pytket-benchmarking-store

Benchpress team @ IBM Quantum

Abdullah Ash Saki

Shelly Garion

Sebastian Brandhofer

Matthew Treinish

Luciano Bello

Ali Javadi

Take aways

The impact of quantum software performance only grows with hardware improvements

No way to decouple quantum hardware performance from that of software for most tests

Quantum SDKs differ markedly in their feature sets, capabilities, and robustness

Benchpress aims to ascertain software performance over multiple frameworks at scales achievable in the next few years

Qiskit outperforms other SDKs in terms of both overall runtime and output 2Q gate counts, and is the only SDK not to fail any tests

SDK performance varies between versions; version information needs to be presented when benchmarking software and hardware

There is no one SDK / recipe that yields best performance across all benchmarks and hardware platforms