TNBS: A Kernel-Based Benchmarking for Digital Quantum Computers

Gonzalo Ferro[1], Oluwatosin Odubanjo[2], Diego Andrade[2(✉)], and Andrés Gómez[1]

[1] Galicia Supercomputing Center (CESGA), Santiago de Compostela, Spain
{gonzalo.ferro.costas,agomez}@cesga.es
[2] CITIC, Computer Architecture Group, Universidade da Coruña, A Coruña, Spain
{oluwatosin.esther.odubanjo,diego.andrade}@udc.es

Abstract. The systematic evaluation of the performance of Quantum Computers allows users to identify the best platform to execute a certain class of workload, and it can also guide the future developments of hardware companies. The NEASQC Benchmark Suite (TNBS) is a methodology designed in the context of the NEASQC (NExt ApplicationS of Quantum Computing) European project to perform such evaluation across several benchmark cases identified as common in one or several domains of application of Quantum Computing. TNBS follows several design principles identified as relevant after a thorough evaluation of the existing benchmarking methodologies: (1) benchmarks are defined at high-level not being linked to any algorithmic approach or implementation, (2) benchmarks are scalable (in qubits) and their output is classically verifiable, (3) performance metrics may be defined per case to fit the nature of the outputs of each case, and (4) the benchmark report also keeps record of all the relevant components of the execution stack. Finally, in the future, the reports may be submitted to a centralized repository, which is equipped with a web interface allowing a systematic and objective comparison of different platforms across the different cases. The first release of TNBS is composed of 4 benchmark cases and provides a reference implementation in myQLM. Documentation and code are available at https://github.com/NEASQC/WP3_Benchmark.

1 Introduction

Digital Quantum Computers are at a level of maturity that permits the execution of quantum circuits that use up to a few hundred qubits. These qubits in the current Noisy Intermediate-Scale Quantum (NISQ) era are prone to quantum decoherence and are delicate to quantum measurements. These issues give rise to the degradation of the accuracy of quantum computations, although they are evolving fast, increasing their capability [1]. Heavy ongoing improvements in qubit count, quality, and stability, targeted at solving this problem, however, have generated diversity in qubit technologies, resulting in multiple computing models, which makes it challenging to design and implement benchmarking

S. Caino-Lores et al. (Eds.): Euro-Par 2024, LNCS 15385, pp. 28–39, 2025.
https://doi.org/10.1007/978-3-031-90200-0_3

methodologies that are detached from this diversity and can appropriately assess the interaction of the components of the Quantum Computer (QC) stack.

Performance characterization protocols for gate-based quantum platforms in the NISQ era study the performance of quantum operations, full applications or synthetic workloads. While application benchmarks can accurately capture the user experience, conducting repetitive performance measurements can be time-consuming; they are also restrictive, as it is harder for the experimenter to isolate the effect on the performance of a specific component of the Quantum Platform (QP) or application. In contrast, quantum synthetic benchmarking offers greater flexibility, as it can be explicitly designed to do so. This lack of restriction may enhance the versatility and precision of benchmarking procedures. The experimenter can isolate and probe specific components or operations of the platform, resulting in a controlled experimentation.

Furthermore, QCs will not work alone. In fact, they can be seen as another accelerator of a complex computing platform, as current GPUs are. For this reason, they are also referred to as Quantum Processing Units (or QPU). In fact, every quantum algorithm is really hybrid, mixing classical and quantum parts. These QPUs can work alone or jointly, defining a Distributed High Performance Quantum Computing infrastructure [2], with or without quantum communications. Any benchmark for digital quantum computers must take into account both the hybrid nature of the tasks and the possibility of using several QPUs.

This paper introduces The NEASQC Benchmark Suite (TNBS) which is based on the analysis of these existing methodologies to propose a full-stack platform-agnostic application-driven benchmarking methodology. The three contributions of this paper are:

1 A discussion of the state of the art of benchmarking of Quantum Computers (Sect. 2).
2 A description of the new NEASQC benchmark suite (TNBS) (Sect. 3).
3 A description of one of its benchmark cases: Probability Loading (PL) (Sect. 4).

2 Review of the State of the Art of Quantum Benchmarking

Quantum Computing benchmarking methodologies usually fall within one of these three categories: Quantum Operations Benchmarking (QOB), Quantum Application Benchmarking (QAB), and Quantum Synthetic Benchmarking (QSB). The evaluation of the reliability of quantum operations in QOB involves techniques of Quantum Process tomography (QPT), Quantum State tomography (QST), Gate set tomography (GST), Direct Fidelity Estimation (DFE), randomized benchmarking (RB), and cycle benchmarking (CB) [5,6] which aim at providing information related to the accuracy, fidelity and effectiveness of gate operations, qubits, quantum states, and quantum processes [9].

Quantum Application Benchmarking (QAB)

QAB relies on the use of practical quantum workloads to compare performance across different quantum platforms and can provide a holistic measure of the computational capabilities of quantum platforms [4]. The designed methodologies include benchmarks for low-level (for example: Quantum Fourier Transform (QFT) and Phase Estimation, which serve as the basic building block of high-level quantum algorithms and applications), high-level (for example: Searching, Hamiltonian Simulation), and hybrid classical-quantum algorithms (for example: Quantum Approximate Optimization Algorithm (QAOA), and Variational Quantum Eigensolver (VQE)). Specifically, low-level algorithms benchmarks are designed to reflect the functions of specific low-level operations within the broader context of an application or an higher level algorithm. In contrast, high-level and hybrid classical-quantum algorithms benchmarking focuses on all aspects of the algorithm.

The SupermarQ suite [3] is an example, which compares the performance of IBM, IonQ, and AQT@LBNL platforms by executing quantum workloads such as QAOA, VQE, Hamiltonian Simulation, etc. Also, QED-C suite [11] highlights quantum workloads such as Hamiltonian simulation, VQE, etc. It also features low-level algorithms such as the QFT, Phase, and Amplitude Estimations. Furthermore, this work (TNBS), is a contribution to low-level algorithms benchmarking.

Quantum Synthetic Benchmarking (QSB)

QSB methodologies can be designed with specially structured and constructed workloads with specific properties to stress certain components of the QC stack. As such, these workloads can be artificially generated or may be inspired by applications. QSB allows for the isolation and controlled experimentation of separated components, and depending on performance assessment, their holistic degree can be influenced by the decision to focus on specific components of the QC stack. In this case, an understanding of individual components of a QP is obtained, but it does not provide a comprehensive assessment of a QPs overall capabilities, leading to a less holistic evaluation. The holistic degree of QSB, can be extended by taking into account the full stack of the QC.

This description of QSB allows us capture it in the context of the structure and specific properties of the workloads as well as its holistic degree which is a measure of controlled experimentation, thereby allowing us to further divide it into Quantum Random Modelled Benchmarking (QRMB) and Quantum Application Modelled Benchmarking (QAMB).

Quantum Random-Modelled Benchmarking (QRMB): This category of QSB characterizes benchmarks that are based on a random model of circuits. Hence, these quantum workloads are artificially generated and are designed to focus on components of the QC stack at the lower level (such as gates, error correction, or noise characteristics) and the compilation level of the stack.

The IBMs Quantum volume (QV) metric evaluation [6], is an example of QRMB that provides a holistic measure [8], of the *degree of generality*[1] of a quantum computer; it takes into account hardware performance (such as fidelity, error rates, crosstalk, etc.), design parameters (such as connectivity of qubits and gate set), and also includes the software for circuit optimization. This benchmark's main drawback is its random-fixed-generic nature of the quantum circuit, non-scalability, and restriction to the square structured quantum circuit model used.

Another example QRMB is the quantum LINPACK [7], which wants to be analogous to the classical LINPACK benchmark. It measures performance of a QC to solve random systems of linear equations, using the RAndom Circuit Block-Encoded Matrix (RACBEM) as a model. Due to the use of a random matrix model and a random quantum circuit model, it does not reflect the behaviour of real-world applications and is also not efficiently scalable.

Quantum Application-Modelled Benchmarking (QAMB): This category of QSB encompasses benchmarks focused on practical quantum workloads that reflect real-world scenarios relevant to various fields. They are based on application-modelled circuits with a systematic structure, aiming to bridge the gap between theoretical capabilities and practical impact. These benchmarks evaluate quantum processors on tasks of practical importance, providing a more holistic approach than QRMBs. While they are not real-world quantum applications themselves, they allow for the extrapolation of likely real-world application performance from the benchmark results and the potential improvement of specific tasks within an application.

As an example, the work of Mills, et al. [4], uses a set of quantum class circuits inspired by structured circuits which reflects practicality in near-term machine learning and chemistry applications, random circuits, product formula circuits, and VQE state preparation circuits. This work takes inspiration from the IBMs quantum volume metric for square modeled circuits and extends to models for deep and shallow quantum circuits reflecting practical significance. The proposed methodology is based on modifying the variable components of the QC stack, therefore, it offers controlled experimentation.

Another proposal that fits into this category is the work of Kobayashi et al., 2022 [10], which allows for a study of the role of optimizers in Variational Quantum Eigensolvers by systematically separating the target Hamiltonian and the ansatz, which are responsible for determining the complexity of the optimization problem in the VQE. With this approach, benchmarks that depend on the structure of the ansatz circuit can be designed.

2.1 Low-Level and High-Level Quantum Benchmarking

Whether a quantum benchmark is low-level or high-level is independent of the choice of quantum circuit design. Rather, it depends on the choice of figures

[1] The degree to which the QC performs as a general purpose QC.

of merits (or metrics). As such, quantum circuits derived randomly or from quantum algorithms or inspired by applications can be used to probe low-level as well as high-level performance information of quantum platforms. In other words, the design of the quantum circuits can influence how effectively it probes certain metrics, but it does not strictly determine the level of the benchmark. In the next section, we briefly look at low-level and high-level quantum benchmarking.

Low-Level Quantum Benchmarking. Low-level benchmarking of quantum computers often focuses on hardware performance information pertaining to error rates, coherence times, qubit connectivity, gate operations, fidelity, etc. As an instance, QASMBench [12] is an open-source application-centric low-level benchmark suite based on the OpenQASM assembly representation; this suite aims at evaluating and characterizing the properties of emerging NISQ devices, estimating the potential of optimization from quantum transpilers, and benchmarking the performance of classical quantum simulators. The authors propose four circuit metrics to assess the execution efficiency, susceptibility to NISQ error, and potential gain from low-level optimizations: gate density, retention lifespan, measurement density, and entanglement variance. Selected test cases are from the domains of chemistry, simulation, linear algebra, machine learning, optimization, etc.

High-Level Quantum Benchmarking. This type of benchmarking abstracts low-level details, but instead focuses on the performance characterization by metrics such as algorithm accuracy, speed in terms of execution times, and resource usage, which is critical in understanding scalability. These metrics in general, give more information about the effectiveness of a quantum computing system in solving particular problems. To illustrate high-level benchmarks, Eviden's universal metric- Q-Score^{TM} [13], designed to measure the actual performance of quantum processors when solving an optimization problem, provides insight into resource usage. The metric measures the maximum number of qubits that can be used effectively to solve the MaxCut combinatorial optimization problem with the QAOA. SupermarQ [3] again, uses a set of feature vectors derived from a combination of high-level (and low-level) metrics such as qubit interactions, gate operations, number of qubits, qubit activity, etc., to estimate the representativeness of chosen applications. Applications are benchmarked using high-level metrics such as the Hellinger distance, and a score obtained by comparing ideal and experimental values.

3 The NEASQC Benchmarking Methodology Proposal

The NEASQC Benchmark Suite (TNBS) is composed of a methodology to define test cases and a set of tests and the rules to execute them and report their results. The NEASQC benchmarking methodology is application driven. It is based on quantum subroutines (or kernels), that are low-level quantum algorithms (such

as state preparation or phase estimation). These subroutines have been extracted from the use cases of QC; generally, they form the basis for high-level quantum applications such as the Shor's algorithm, VQE, etc.

The methodology has been designed taking into account the next objectives:
Objective 1: The benchmark suite must be representative. This is enabled by the selection of benchmark cases that are extracted from workloads commonly executed in quantum platforms.
Objective 2: The methodology has to holistically and procedurally characterize the performance of all the components of the execution stack. This implies (1) the design of a benchmark case definition template, (2) the design of formal procedures for the execution of the benchmarks and results collection, and (3) the selection of relevant metrics for evaluating and comparing different platforms.
Objective 3: The methodology must be resilient to near and mid-term technological changes in quantum technology. This implies that each benchmark may be proposed for a different number of qubits and their results should be verifiable classically in all cases. Also, it must permit the execution on distributed platforms.
Objective 4: The methodology must help the developers and manufactures to improve their solutions from application layer to hardware devices. As such, the methodology must be agnostic to the hardware and software, defining each case from mathematical or procedure descriptions.

The rest of this section starts with the elements required to define a benchmark test case (Sect. 3.1) and a description of general aspects of the evaluation metrics 3.2.

3.1 Benchmark Case Definition

A core component of the TNBS methodology is a well-defined template for the specification of the benchmark cases. The main components of this template are: the kernel, the benchmark test case, the execution procedure and the example implementation.

Kernel. A *kernel* is a core *quantum subroutine* common to several quantum applications. From a computational perspective, a kernel is a task or routine to be run on a quantum device. Hence, it should have some part that should be implemented as a (or several) quantum circuit(s). Examples are quantum probability loading and quantum phase estimation.

In TNBS, kernel specification is done mathematically or procedurally. This definition style ensures that the kernel can be implemented at will, not being necessarily linked to a given algorithmic approach or implementation which may favour some quantum platforms over others introducing unwanted bias in the evaluation.

Benchmark Test Case. A *Benchmark Test Case (BTC)* is a scalable application (in the number of qubits) restricted only to evaluating the kernel associated

with a given benchmark case. The BTC cannot involve interaction with other classical routines, as is the case of classical optimizers in variational algorithms (but it can include classical sections to prepare data or postprocess quantum results to return a valid result). A BTC is designed to be verifiable easily by classical means and is also suitable for execution in QPUs of the NISQ era. Although, it should be defined to be used with fault-tolerant QPUs as well.

In addition to time-performance measurements, TNBS is designed to use kernel-dependent metrics to measure the accuracy of the outputs generated by the BTC. The model of verification is a classical/analytical model, and because verification is kernel-dependent, it may be different for each benchmark test case.

As a consequence, the *performance* of a BTC on a quantum platform is characterized by two metrics: accuracy and speed. The *accuracy* of the BTC is verified against a ground-truth obtained from a classical calculation or an analytical method; the accuracy metrics are usually tailored for each kernel. *Speed* measures the execution time of the BTC.

Execution Procedure. The execution of a BTC must follow a well-specified protocol which may be different for each benchmark case. This procedure is usually composed of:

1. A preparation phase, where some preprocessing of the data may be done before inputs are sent to the quantum circuit. This preparation takes place in a classical CPU.
2. The execution of the quantum circuit, which takes place in the QPU (or in several QPUs). This is the focus of our study.
3. A post-processing phase, where the raw outputs of the quantum circuit may be transformed to produce the required output.

This execution procedure should realize the execution of the BTC for circuits of different number of qubits. Also, it calculates the number of shots of the quantum circuits to generate results within a given confidence interval and the number of repetitions of the procedure to guarantee that the results are later verifiable and comparable.

Implementation. By definition, TNBS benchmark cases do not have an official implementation. Still, we intend to provide a sample myQLM-compatible implementation for each benchmark case, just for illustration purposes. The mathematical or algorithmic definition of the kernels is motivated by our thesis that linking a case to a given implementation, or even an algorithmic approach, can bias the case and favor some platforms over others.

3.2 Evaluation Metrics

In this section, we further discuss the metrics used for the evaluation of the BTC. As mentioned above, these metrics evaluate two different aspects: the accuracy of the computation and its speed.

Accuracy Metrics. The accuracy metrics must measure the quality of the generated output against a classically-generated ground-truth. As the outputs generated may be different for each specific problem, these metrics are usually defined per case, and even more, each case can have several metrics. Examples of such metrics are the Kolmogorov-Smirnov (KS) and Kullback-Liebler (KL) divergence used in the Probability Loading BTC.

Speed Metrics. Speed is evaluated through a timing analysis. This analysis is carried out through the execution of the case following the execution procedure. Because of the hybrid nature of quantum computing (and even the existence of hybrid solutions for some algorithms), TNBS BTCs must measure separately the execution time of the classical preprocessing and postprocessing phases and the quantum demands. This is why the timing is decomposed into three components: (1) The wall time, which is the total execution time of the complete execution procedure of the BTC, (2) The quantum time, which is the execution time of the quantum circuit(s), and (3) The classical time, which is the execution time of the non-quantum parts of the execution procedure (pre and post-processing tasks).

These metrics permit the identification of the bottlenecks (if quantum or classical) and can cope with future algorithms that permit the interaction of quantum and classical codes wisely.

Result Reporting. The results of each benchmark case on a given platform are reported using a JSON file which contains the specification of the relevant components of the stack, other relevant information such as the date and time where the experiment took place, and the outputs generated for each case. A JSON schema is provided to allow the executors to generate the output and verify their correctness before submission to the TNBS centralized repository of results. This repository will have a website that allows the comparison of the benchmark results of different platforms.

4 Probability Loading: A Sample Benchmark Case

In this section, we provide, as an example of a benchmark case, a short description of the Probability Loading (PL) use case of the TNBS. First of all, the *PL kernel* has been selected as a benchmark case because it is a building block of several quantum algorithms like the Harrow-Hassidim-Lloyd (*HHL*) [14], the quantum principal component analysis (*PCA*) [15], the quantum amplitude estimation algorithms, etc. In these algorithms, the *PL kernel* is usually used at the initialization part of the circuit to load a probability distribution in a set of qubits. This kernel is computationally intensive because its number of operations typically scales as $\sim 2^n$, n being the number of qubits to be initialized.

In the following sections, we explain in turn, the kernel specification, the associated BTC and how its performance is characterized.

4.1 Kernel Specification

As we established in the previous section, in TNBS, kernels are defined mathematically or algorithmically, and this is the kind of definition that we are giving for the *PL kernel*. Given a list of normalized vectors,

$$\mathbf{V} = \{v_0, v_1, \cdot, v_{2^n-1}\}, v_i \in \mathbb{C}^{2^n}; \sum_{i=0}^{2^n-1} |v_i|^2 = 1 \tag{1}$$

The main task of the *PL kernel* is to generate an operator $\mathbf{U}$, from the normalised vector $\mathbf{V}$, which satisfies the equation:

$$\mathbf{U}|0\rangle_n = \sum_{i=0}^{2^n-1} v_i|i\rangle_n \tag{2}$$

Hence, this kernel can be used to load a fixed normal probability density function, $N_{\mu,x}$. Such that, the Eq. 1 can be re-expressed as:

$$\mathbf{P} = \{p_0, p_1, \cdot, p_{2^n-1}\}, p_i \in [0, 1]; \sum_{i=0}^{2^n-1} |p_i|^2 = 1 \tag{3}$$

Thus, the Eq. 2 becomes:

$$\mathbf{U}|0\rangle_n = \sum_{i=0}^{2^n-1} \sqrt{p_i}|i\rangle_n \tag{4}$$

4.2 PL Benchmark Test Case

In the TNBS methodology, the performance of the kernel must be evaluated in the context of a BTC. The BTC of the PL kernel consists of loading a probability distribution in a set of qubits; this BTC can be defined for an arbitrary number of qubits, n, and its output can be compared to the actual probability distribution. The required operator, U, is obtained by encoding a list of normalized probabilities, $P_{norm}(x)$, into a n-qubits quantum circuit. This list of normalized probabilities, $P_{norm}(x)$, is created from an array, x, which is bounded by a fixed mean, $\tilde{\mu}$ and fixed standard deviation, $\tilde{\sigma}$. This list can feed different *PL* algorithms from the literature prepared to generate the corresponding U operator (check one of them in [16]).

The quantum operation, $U|0\rangle_n$ in Eq. 4 is executed, measuring the output qubits a numer of times given by:

$$n_{shots} = min\left(10^6, \frac{100}{min(P_{norm}(x_i)}\right) \tag{5}$$

These measurements reconstruct a probability distribution, Q. And the reconstruction is done by calculating the ratio of times in which a certain state $|i\rangle_n$ is obtained from n_{shots}.

4.3 Performance

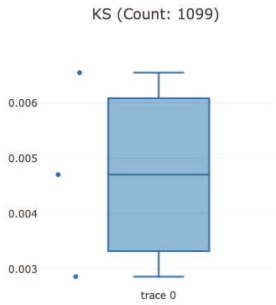
The accuracy of the outputs of the BTC is calculated by comparing the values of the measured probability distribution, Q with those of the actual normalized probability distribution, P_{norm}, using two metrics:

– *The Kolmogorov-Smirnov (KS)*, between Q and P_{norm}.

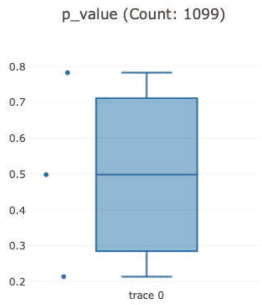
$$KS = \max\left(\left|\sum_{j=0}^{i} P_{norm}(x_j) - \sum_{j=0}^{i} Q_j\right|, \ \forall i = 0, 1, \cdots, 2^n - 1\right) \tag{6}$$

– and *The Kullback-Leibler divergence (KL).*

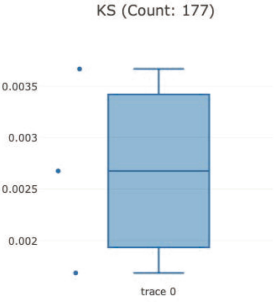
$$KL(Q/P_{norm}) = \sum_{j=0}^{2^n-1} P_{norm}(x_j) \ln \frac{P_{norm}(x_j)}{\max(\epsilon, Q_k)} \tag{7}$$



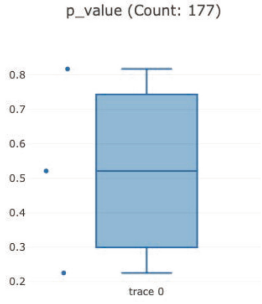
(a) KS: 4 qubits



(b) p-value: 4 qubits



(c) KS: 6 qubits



(d) p-value: 6 qubits

Fig. 1. KS and p-values obtained in ideal simulations for 4 and 6 qubits using 1099 and 177 shots respectively

where $\epsilon = \min(P_{norm}(x_j)) * 10^{-5}$, which guarantees the logarithm exists when $Q_k = 0$

Also, the Chi- Square test, χ^2, generates a *p-value* from $n_{shots}Q$ and $n_{shots}P_{norm}$. If this *p-value* is lower than *0.05*, the output of the BTC is labelled as not valid. In addition, the speed of a platform to execute the BTC is obtained by measuring the elapsed and the quantum execution times of the BTC. As a sample test, Figs. 1(a), b c, (d) are boxplots of the *KS* and *p-values* obtained for 4 and 6 qubits respectively on the myQLM ideal simulator.

5 Conclusions

The NEASQC Benchmark Suite (TNBS) proposes a methodology based on kernels, i.e., subroutines common to several user-level applications. Once the kernels are identified, they are described mathematically or procedurally, allowing the implementation of different versions depending on the advance of the software or the specific hardware characteristics. To evaluate the performance and quality of the implementation, one test case, coined as Benchmark Test Case (BTC), accompanies the kernel definition. This case is defined in such a way that it can be verified easily using classical or analytical procedures, it must scale up to a reasonable number of qubits, and it must include metrics to evaluate the quality of the results. The Probability Loading (PL) benchmark case is provided as a benchmark case example. As with any other benchmark, the performance is measured, but in addition to the total wall-time, the times spent in the CPUs and the QPUs are also reported separately.

Acknowledgments. All authors acknowledge the European Project NExt ApplicationS of Quantum Computing (NEASQC), funded by Horizon 2020 Program inside the call H2020-FETFLAG-2020-01 (Grant Agreement 951821). Grant PID2022-136435NB-I00, funded by MCIN/AEI/10.13039/501100011033 and by "ERDF A way of making Europe", EU. Grant for the Predoctoral contract of O.E. Odubanjo ref. PREP2022-000281 funded by MCIN/AEI/10.13039/501100 011033 and by ESF+. The authors also acknowledge Galicia Supercomputing Center (CESGA) for providing access to Finis-Terrae III supercomputer with financing from the Programa Operativo Plurirregional de Espaa 2014-2020 of ERDF, ICTS-2019-02-CESGA-3. CITIC, as a center accredited for excellence within the Galician University System and a member of the CIGUS Network, receives subsidies from the Department of Education, Science, Universities, and Vocational Training of the Xunta de Galicia. Additionally, it is co-financed by the EU through the FEDER Galicia 2021-27 operational program (Ref. ED431G 2023/01).

References

1. Robledo-Moreno, J., et al.: Chemistry Beyond Exact Solutions on a Quantum-Centric Supercomputer. arXiv:2405.05068 (2024)
2. Barral, D., et al.: Review of Distributed Quantum Computing. From single QPU to High Performance Quantum Computing. arXiv:2404.01265 (2024)

3. Tomesh, T., et al.: SupermarQ: a scalable quantum benchmark suite. In: 2022 IEEE International Symposium on HPCA, pp. 587–603 (2022)
4. Mills, D., Sivarajah, S., Scholten, T.L., Duncan, R.: Application-motivated, holistic benchmarking of a full quantum computing stack. Quantum **5**, 415 (2021)
5. Resch S., J., Karpuzcu, U.R.: Benchmarking quantum computers and the impact of quantum noise. ACM Comput. Surv. **54**(7), 1–35 (2021)
6. Cross, A.W., J., Bishop, L.S., Sheldon, S., Nation, P.D.: Validating quantum computers using randomized model circuits. Phys. Rev. A **100**(032328) (2019)
7. Dong, Y., Lin, L.: Random circuit block-encoded matrix and a proposal of quantum linpack benchmark. Phys. Rev. A **103**(062412) (2021)
8. Wack, A., et al.: Quality, Speed and Scale: three key attributes to measure the performance of near-term quantum computers. arXiv:2110.14108 (2021)
9. Chuang, I.N., Nielsen, M.A.: Prescription for experimental determination of the dynamics of a quantum black box. J. Mod. Opt. **44**(11–12), 2455–2467 (1997)
10. Kobayashi, F., Mitarai, K., Fujii, K.: Parent Hamiltonian as a benchmark problem for variational quantum eigensolvers. Phys. Rev. A **105**(052415), (2022)
11. Lubinski, T., et al.: Application-Oriented Performance Benchmarks for Quantum Computing. arxiv:2110.03137 (2023)
12. Li, A., Stein, S., Krishnamoorthy, S., Ang, J.: QASMBench: a low-level quantum benchmark suite for NISQ evaluation and simulation. ACM Trans. Quantum Comput. **4**(2) (2023)
13. Martiel, S., Ayral, T., Allouche, C.: Benchmarking quantum coprocessors in an application-centric, hardware-agnostic, and scalable way. IEEE Trans. Quant. Eng., 1 (2021)
14. Harrow, A.W., Hassidim, A., Seth, L.: Quantum algorithm for linear systems of equations. Phys. Rev. Lett. **103**(15) (2009)
15. Lloyd, S., Mohseni, M., Rebentrost, P.: Quantum principal component analysis. Nat. Phys. **10**(9), 631–633 (2014)
16. Shende, V.V., Bullock, S.S., Markov, I.L.: Synthesis of quantum-logic circuits. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **25**(6), 1000–1010 (2006)