qWard: A Unified Toolkit for Pre- and Post-Runtime Quantum Circuit Metrics

1st Cristian Marquez
Department of Systems and Computing Engineering
Universidad de los Andes
Bogotá, Colombia
c.marquezb@uniandes.edu.co

2nd Daniel Sierra
Electrical Engineering and Computer Science
The Catholic University of America
Washington DC, 20064
sierrasosa@cua.edu

3rd Kelly Garcés
Department of Systems and Computing Engineering
Universidad de los Andes
Bogotá, Colombia
kj.garces971@uniandes.edu.co

***Abstract*—As quantum computing (QC) matures towards practical applications, effective methods to evaluate quantum algorithms, circuits, and execution quality in quantum software are gaining more relevance. To apply such methods, a set of meaningful metrics must be defined, understood, calculated, and applied. In an effort to solve that problem, this paper categorizes quantum related metrics into pre-runtime (derived from the static analysis of circuits before execution) and post-runtime (derived from the analysis of execution results). It is worthwhile to mention that while popular quantum SDKs such as Qiskit, PennyLane, and Q# offer some circuit characteristics, there is often a gap between these native capabilities and the broader set of insightful pre- and post-runtime metrics in the literature. This paper introduces *qWard*, a Python library developed to bridge this gap by offering an initial implementation for the Qiskit SDK. *qWard* provides developers and researchers with a toolkit to calculate and visualize key pre-runtime and post-runtime metrics, facilitating the analysis of quantum circuits quality. The scope of this first qWard version is to provide support for the Qiskit SDK and the Qiskit AER simulator.**

***Index Terms*—Quantum computing, software quality, open source library, Qiskit**

I. Introduction

QC is an emerging field based on quantum mechanics principles [1], enabling systems to leverage phenomena such as superposition and entanglement. This novel approach has the potential to address computational challenges currently impossible for classical hardware, promising advances in areas such as large-scale number factorization [2], cryptography [3], and complex model training [4], [5].

To effectively evaluate the overall quality of any given software asset, the use of metrics is essential. In the context of QC, the development process typically involves two global steps: first, developers prepare quantum states by applying gates and planning measurements, and second, execute the circuit to collect results. This workflow, which includes some intermediate steps such as transpilation, calibration, and optimization before actual execution, naturally lends itself to temporal metrics categorization. Consequently, this paper adopts the terms **pre-runtime metrics** for those measures computed after coding the algorithm but before its execution (e.g., number of gates, circuit depth and width) and **post-runtime metrics** for those derived from the execution (e.g., success rates, fidelity, execution time).

The literature identifies various categories of pre-runtime metrics. These metrics often focus on *maintenance aspects*, for instance, the complexity of gate operations, which cover metrics such as the number of qubits and the number of 2-gate operations [6]–[8]. Additionally, they address *circuit complexity aspects*, which includes structural attributes such as circuit width, depth, and gate complexity [8].

Complementing the pre-runtime perspective, post-runtime metrics commonly analyze *performance-related aspects* like execution time, precision, and accuracy [6], [9], *hardware quality aspects* such as qubit coherence times (T1, T2) and gate error rates [10], and *correctness and error aspects*, often evaluating circuit fidelity or success rates, for instance, through protocols like randomized benchmarking [11].

The current study is motivated by the need for a tool to collect and analyze a wide spectrum of metrics. This motivation leads to the following research questions.

RQ1 : To what extent do existing QC SDKs facilitate the collection of pre- metrics and the calculation of post-runtime metrics?
RQ2 : Do QC SDKs effectively incorporate emerging metrics as reported in the literature?
RQ3 : How can a library be designed to address the need to collect and analyze a wide spectrum of quantum circuit metrics?

II. qWard

qWard is a Python library, designed to assist developers and researchers in analyzing quantum circuit quality based on the aforementioned metrics. It offers features to **calculate** and **visualize** metrics through graphs, thus facilitating informed decision-making during the development phase and supporting the comprehensive evaluation of quantum programs.

For in-depth reference and implementation details, extensive documentation of the API and more elaborate diagrams are available in the official documentation repository https://xthecapx.github.io/qiskit-qward/. Additionally, the alpha version can be accessed via the Python package manager PIP at the following URL https://pypi.org/project/qiskit-qward/.

III. Discussion

In relation to RQ1, our research indicates that current QC SDKs offer insufficient support for detailed circuit analysis. In particular, after experimenting with Qiskit in the initial version of qWard, we notice that the library provides fundamental circuit characteristics such as depth, width, and gate counts, and it also provides basic tools for understanding the execution results like the histogram plot. However, it does not implement the advanced metrics that are extensively covered in the academic literature and lacks robust tools for execution analysis.

With reference to RQ2, it is evident that existing SDKs do not integrate emerging metrics from recent publications. The disparity between theoretical advances and practical implementation forces researchers to code the same analysis repeatedly, resulting in inconsistent definitions of metrics and varying calculation methodologies.

Finally, to answer RQ3, qWard shows that a metrics library can be effectively designed by implementing the strategy pattern for consistent metrics integration, using Pydantic for schema validation, and including visualization tools to contribute to the analysis. We also conclude that documentation is an essential part of these kinds of library, as it is crucial to inform the user about the related work supporting the metrics included on each strategy.

IV. Conclusions

As QC rapidly progresses while also facing significant challenges such as high error rates, substantial costs, and limited access, comprehensive metric analysis becomes increasingly vital. This paper began by conducting a quick literature review that identified key pre-runtime and post-runtime metrics for evaluating quantum circuits. The review made evident the gap between the properties available in current SDKs (e.g., Qiskit, PennyLane, or Q#) and the metrics reported in the literature. These findings led to the conclusion that a unified tool is needed to calculate a broad spectrum of such metrics.

To bridge this identified gap, the qWard library was introduced. qWard was designed to assist researchers and developers in the systematic collection and analysis of quantum circuit metrics. The purpose of the library is to empower users to make better informed decisions during the quantum software development process, facilitate benchmark preparations, and improve understanding of quantum programs.

Ultimately, qWard seeks to contribute to the maturation of the quantum software ecosystem by promoting a data-driven approach to circuit design and execution analysis. This involves supporting the understanding and visualization of resource utilization and, thereby, improving quantum computations.

A. Future Work

Looking ahead, the development of qWard is an ongoing effort of the TICSw research group of the Universidad de los Andes de Colombia, with several features envisioned to enhance its capabilities, including the following:

Expansion of Metrics Library and Community Contributions: As an open source project, qWard actively seeks community contributions. We envision qWard becoming a collaborative platform for the quantum computing community. Researchers developing novel pre-runtime or post-runtime metrics are invited to integrate their findings into the library's codebase.

Support for Additional Quantum SDKs: While currently focused on Qiskit, a long-term goal is to extend qWard's core functionalities to support other quantum programming SDKs, potentially transforming qWard into an extensible quantum metrics analysis platform.

Advanced Correlation Analysis: Once a comprehensive set of metrics is established, the next step involves analyzing the correlations between pre- and post-runtime metrics. Based on these results, we envision developing models to predict post-runtime metrics from a footprint vector of pre-runtime characteristics.

Visualization and Reporting Enhancements: As part of the beta version, we plan to increase qWard's built-in visualization capabilities and explore integrations with common data analysis and reporting tools.

References

[1] D. Maslov, N. Yunseong, and J. Kim, "An outlook for quantum computing," 2019. [Online]. Available: http://www.ieee.org/publications_standards/publications/rights/index.html

[2] D. Willsch, M. Willsch, F. Jin, H. D. Raedt, and K. Michielsen, "Large-scale simulation of shor's quantum factoring algorithm," *Mathematics*, vol. 11, no. 19, 2023. [Online]. Available: https://www.mdpi.com/2227-7390/11/19/4222

[3] L. Upadhyay, "Quantum cryptography: A survey," *Advances in Intelligent Systems and Computing*, vol. 939, pp. 20–35, 2019. [Online]. Available: https://link-springer-com.ezproxy.uniandes.edu.co/chapter/10.1007/978-3-030-16681-6_3

[4] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature Publishing Group*, vol. 549, 2017.

[5] S. Garg and G. Ramakrishnan, "Advances in quantum deep learning: An overview," May 2020. [Online]. Available: https://arxiv.org/abs/2005.04316v1

[6] E. Moguel, J. Rojo, D. Valencia, J. Berrocal, J. Garcia-Alonso, and J. M. Murillo, "Quantum service-oriented computing: current landscape and challenges," *Software Quality Journal*, vol. 30, pp. 983–1002, 12 2022. [Online]. Available: https://link.springer.com/article/10.1007/s11219-022-09589-y

[7] J. Alvarado-Valiente, J. Romero-Álvarez, A. Díaz, M. Rodríguez, I. García-Rodríguez, E. Moguel, J. Garcia-Alonso, and J. M. Murillo, "Quantum services generation and deployment process: A quality-oriented approach," *Communications in Computer and Information Science*, vol. 1871 CCIS, pp. 200–214, 2023. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-031-43703-8_15

[8] A. Díaz, J. Alvarado-Valiente, J. Romero-Álvarez, E. Moguel, J. Garcia-Alonso, M. Rodríguez, I. García-Rodríguez, and J. M. Murillo, "Service engineering for quantum computing: Ensuring high-quality quantum services," *Information and Software Technology*, vol. 179, p. 107643, Mar. 2024.

[9] J. Verduro, M. Rodríguez, and M. Piattini, "Software quality issues in quantum information systems," *Q-SET@QCE*, 2021.

[10] M. Alam, A. Ash-Saki, and S. Ghosh, "Addressing temporal variations in qubit quality metrics for parameterized quantum circuits," *Proceedings of the International Symposium on Low Power Electronics and Design*, vol. 2019-July, 7 2019.

[11] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, "Randomized benchmarking of quantum gates," *Physical Review A*, vol. 77, no. 1, p. 012307, Jan. 2008, arXiv:0707.0963 [quant-ph].