PurpleCaffeine: tracking of quantum programs and experiments

Iskandar Sitdikov [1¶], **Michaël Rollin** [2], **Ansah Mohammad Kuriyodath** [3], **and Luis Eduardo Martínez Hérnandez** [4]

1 IBM Quantum, T.J. Watson Research Center, Yorktown Heights, NY 10598, USA **2** Shape-IT, France **3** Sardar Vallabhbhai National Institute of Technology, Surat, India **4** Netcracker, Mexico ¶ Corresponding author

DOI: 10.21105/joss.05673

Software

- Review
- Repository
- Archive

Editor: Lucy Whalley

Reviewers:

- @DevenPanchal
- @obliviateandsurrender

Submitted: 19 June 2023
Published: 02 October 2023

License
Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License (CC BY 4.0).

Summary

PurpleCaffeine aims to provide researchers in the field of quantum computing with a user-friendly and efficient solution for tracking their experimental data. With the rapid advancement of quantum computing research, the need for accessible and organized data management tools has become increasingly important. By offering a simple interface, PurpleCaffeine allows researchers to easily record and organize quantum experimental data, ensuring its accessibility and facilitating future analysis.

Researchers can use PurpleCaffeine to capture and store crucial information related to their quantum experiments. The user-friendly interface simplifies the process of inputting and organizing data, including experimental parameters, measurement results, quantum circuits, OpenQASM (Cross et al., 2017) files, devices information and other relevant metadata. The emphasis on simplicity reduces the learning curve and frees researchers from complex data management tasks, enabling them to focus on their core work.

Statement of need

Researchers in the field of quantum computing rely predominantly on notebook services, such as Jupyter (Kluyver et al., 2016), to work within an interactive coding environment. While this approach offers numerous benefits, including code experimentation and real-time analysis, it presents a significant challenge when it comes to tracking experimental data. One of the main drawbacks is the constant overwriting of data, making it exceedingly difficult to trace the specific parameters, circuits, and other details used in previous iterations.

As quantum computing research progresses it becomes increasingly important to have a reliable and efficient system for managing experimental data. A comprehensive solution is needed to address the shortcomings of interactive workflows, ensuring that researchers can easily record and access vital information related to their experiments. By overcoming the limitations of current approaches, this solution empowers researchers to track their data effectively and gain valuable insights from previous iterations, leading to more accurate analyses, reproducible research, and accelerated progress in the field.

Architecture

The proposed software package uses a variation of the client-server architecture, consisting of a client-side component and multiple storage options for storing experimental data Figure 1.

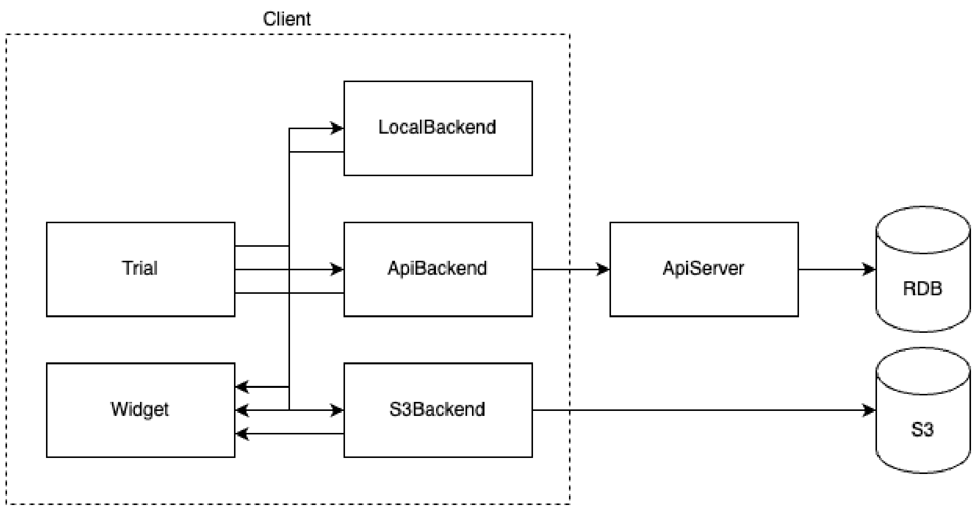


Figure 1: Architecture.

The client component is a Python library specifically designed for tracking experimental data in quantum computing research, covering essential Qiskit (Qiskit contributors, 2023) objects including QuantumCircuit, Operators and Backends.

To cater to diverse needs, the package provides three flavors of storage options for storing experimental data:

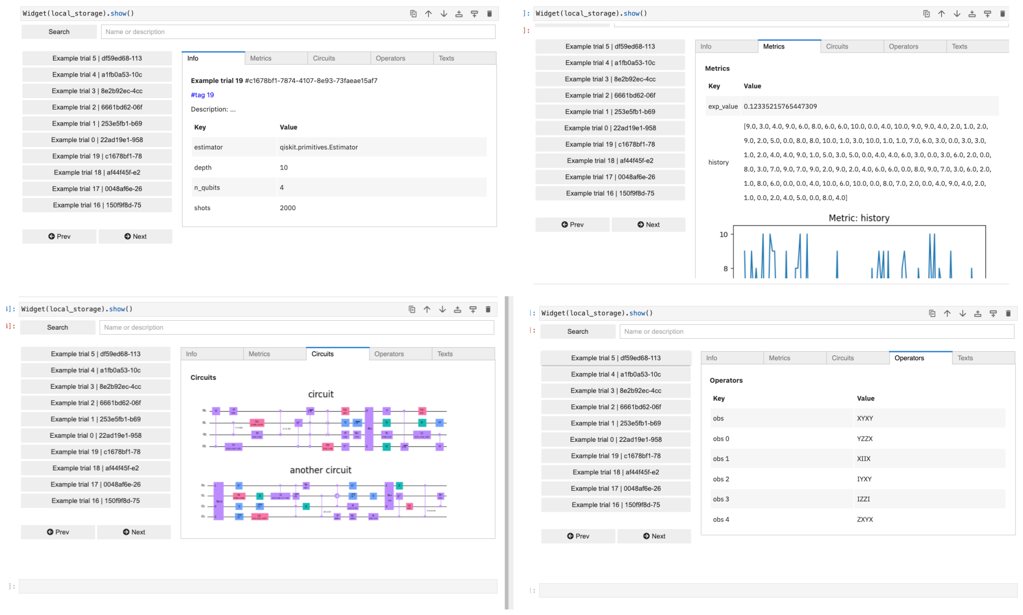
1. Local Storage: The local storage allows researchers to store their experimental data locally on their machines. This option offers convenience and data privacy, as researchers have direct control over their data storage. It is an excellent choice for individual researchers or small-scale projects where data sharing and collaboration are not a primary concern.
2. API Storage: The API storage is a RESTful API service that functions as a multi-tenant storage solution. Researchers can utilize this storage to store their experimental data in a centralized and scalable manner. It is particularly beneficial for collaborative research projects or environments where data sharing and team collaboration are essential.
3. S3 Storage: The S3 storage provides the option to store experimental data in S3 buckets, which are highly scalable and reliable storage containers. Researchers can leverage the power and versatility of S3 to securely store and access their experimental data. This option is ideal for projects that require large-scale data storage and long-term data retention.

Functionality

The functionality of the software package revolves around the Python client, which serves as the primary interface for interacting with the software. Through the client, researchers can access and utilize three key abstractions that cover the entire experimentation tracking workflow:

1. Trial: The Trial class is a fundamental abstraction provided by the software. Researchers use this class to define and specify the experimental metadata they want to gather and track. It acts as a blueprint for capturing information such as experimental parameters, circuit configurations, measurement results, and any other relevant data points.

2. Storage: The Storage abstraction is responsible for the storage functionality of the software. It provides a unified interface to store and retrieve experimental data. The software offers three different flavors of storage options, as previously discussed: local storage, API storage, and S3 storage.
3. Widget: The Widget Figure 2 is a visualization tool that enhances the user experience by acting as a user interface for searching, viewing, and analyzing previous trials. This widget is specifically designed to integrate with Jupyter notebooks, providing a convenient and interactive environment for researchers.



```
Widget(local_storage).show()
```

Search | Name or description

Example trial 5 | df59ed68-113
Example trial 4 | a1fb0a53-10c
Example trial 3 | 8e2b92ec-4cc
Example trial 2 | 6661bd62-06f
Example trial 1 | 253e5fb1-b69
Example trial 0 | 22ad19e1-958
Example trial 19 | c1678bf1-78
Example trial 18 | af44f45f-e2
Example trial 17 | 0048af6e-26
Example trial 16 | 150f9f8d-75

Prev | Next

Info | Metrics | Circuits | Operators | Texts

Example trial 19 #c1678bf1-7874-4107-8e93-73faeae15af7

#tag 19

Description: ...

Key	Value
estimator	qiskit.primitives.Estimator
depth	10
n_qubits	4
shots	2000

Metrics

Key	Value
exp_value	0.12335215765447309
history	[9.0, 3.0, 4.0, 9.0, 6.0, 8.0, 6.0, 6.0, 10.0, 0.0, 4.0, 10.0, 9.0, 9.0, 4.0, 2.0, 1.0, 2.0, 9.0, 2.0, 5.0, 0.0, 8.0, 8.0, 10.0, 1.0, 3.0, 10.0, 1.0, 1.0, 7.0, 6.0, 3.0, 0.0, 3.0, 3.0, 1.0, 2.0, 4.0, 4.0, 9.0, 1.0, 5.0, 3.0, 5.0, 0.0, 4.0, 4.0, 6.0, 3.0, 0.0, 3.0, 6.0, 2.0, 0.0, 8.0, 3.0, 7.0, 9.0, 7.0, 9.0, 2.0, 9.0, 2.0, 4.0, 6.0, 6.0, 0.0, 8.0, 9.0, 7.0, 3.0, 6.0, 2.0, 1.0, 8.0, 6.0, 0.0, 0.0, 4.0, 10.0, 6.0, 10.0, 0.0, 8.0, 7.0, 2.0, 0.0, 4.0, 9.0, 4.0, 2.0, 1.0, 0.0, 2.0, 4.0, 5.0, 0.0, 8.0, 4.0]

Circuits

Operators

Key	Value
obs	XYXY
obs 0	YZZX
obs 1	XIIX
obs 2	IYXY
obs 3	IZZI
obs 4	ZXYX

Figure 2: Widget.

Serialization

`Trials` are serialized using Python's `json` module. We've implemented custom `json.JSONEncoder` and `json.JSONDecoder` to handle quantum computing objects like `QuantumCircuit`, `Backend`, `Operator`, etc.

Encoders and decoders are using `pickle` module to serialize some classes (`Backend`). The pickle module is not secure. Only unpickle data you trust. It is possible to construct malicious pickle data which will execute arbitrary code during unpickling. Never unpickle data that could have come from an untrusted source, or that could have been tampered with.

Acknowledgements

We would like to express our sincere gratitude to the Qiskit Advocates Mentorship Program (QAMP) ("Qiskit Advocate Mentorship Program," 2023) that played a role in bringing together the talented team behind the development of this software package. The mentorship program provided a valuable platform for fostering collaboration, learning, and professional growth.

References

Cross, A. W., Bishop, L. S., Smolin, J. A., & Gambetta, J. M. (2017). *Open quantum assembly language*. https://arxiv.org/abs/1707.03429

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., & Willing, C. (2016). *Jupyter notebooks – a publishing format for reproducible computational workflows* (F. Loizides & B. Schmidt, Eds.; pp. 87–90). IOS Press.

Qiskit advocate mentorship program. (2023). In *GitHub repository*. GitHub. https://github.com/qiskit-advocate/qamp-spring-23

Qiskit contributors. (2023). *Qiskit: An open-source framework for quantum computing*. https://doi.org/10.5281/zenodo.2573505